UNIVERSIDADE DO VALE DO RIO DOS SINOS - UNISINOS
UNIDADE ACADÊMICA DE GRADUAÇÃO
CURSO DE ENGENHARIA ELÉTRICA

LUCAS DE LIMA PINHEIRO

DESENVOLVIMENTO DE UM SUPERVISÓRIO PARA MONITORAMENTO E PROCESSAMENTO DE SINAIS EMG E ACC

São Leopoldo
2023

LUCAS DE LIMA PINHEIRO

DESENVOLVIMENTO DE UM SUPERVISÓRIO PARA MONITORAMENTO E PROCESSAMENTO DE SINAIS EMG E ACC

Trabalho de Conclusão de Curso apresentado como requisito parcial para obtenção do título de Bacharel em Engenharia Elétrica, pelo Curso de Engenharia Elétrica da Universidade do Vale do Rio dos Sinos (UNISINOS).

Orientador: Prof. Dr. João Olegário de Oliveira de Souza

São Leopoldo

2023

RESUMO

Com o atual cenário da eletrônica e computação, estudos envolvendo soluções para pessoas com deficiência física tem se tornado cada vez mais frequentes, como é o caso das tecnologias vestíveis, que realizam a aquisição de sinais corporais, e próteses inteligentes gerenciadas por sinais musculares. Com a ideia de estudar a área de tecnologia, este trabalho apresenta o desenvolvimento de um supervisório para PC que é capaz de coletar e armazenar sinais de eletromiografia (EMG) e sinais de acelerometria (ACC). Os sinais são coletados através de uma plataforma BITalino, a aquisição dos sinais EMG se dá por meio de eletrodos posicionados no braço, permitindo a captação dos sinais corporais relacionados aos movimentos musculares, já o sinal ACC é obtido por um sensor posicionado próximo ao corpo, permitindo a detecção dos movimentos realizados. O supervisório para PC foi desenvolvido utilizando a linguagem Python e a interface gráfica foi implementada com o auxílio do software Qt Designer, foram coletados os sinais através dos movimentos das mãos para quatro sinais analógicos EMG (Eletromiografia) e também através da movimentação do sensor para um canal analógico ACC (acelerometria). Para o sistema foi utilizado uma taxa de amostragem de 1 kHz, o sistema apresenta funções gráficas que exibem os sinais em função da quantidade de amostras e sua amplitude em milivolts, tendo também a opção de selecionar quais dos sinais coletados deseja-se exibir no supervisório para PC, podendo selecionar até quatro opções de gráficos para visualização no supervisório.

Palavras-chave: Sinal de acelerometria. Sinal de Eletromiografia. Supervisório.

LISTA DE FIGURAS

LISTA DE SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
API	Interface de Programação de Aplicativos (*Application Programming Interface*
ACC	Acelerometria (*Accelerometry*)
ECG	Eletrocardiografia (*Electrocardiography*)
EDA	Atividade Eletrodérmica (*Electrodermal Activity*)
EEG	Eletroencefalografia (*Eletroencephalography*)
EMG	Eletromiografia (*Electromyography*)
EOG	Eletrooculograma (*Electrooculogram*)
FPGA	Matriz de portas programáveis em campo (*field programmable gate array)*
IEEE	Instituto de engenheiros eletricista e eletrônicos (*Institute of Electrical and Electronics Engineers*)
MEMS	Sistemas microeletromecânicos (*micro-electro-mechanical systems*)
MUAP	Potencial de ação da unidade motora (*Motor Unit Action Potencial*)
NBR	Normas Brasileiras de Regulação
PC	Computador pessoal (*Personal computer*)
Wifi	Rede local sem fios
WPAN	Rede de área pessoal sem fio (*Wireless Personal Area Network*)

SUMÁRIO

1 INTRODUÇÃO

Com o avanço contínuo da tecnologia, eletrônica e computação nas últimas décadas, os circuitos integrados e microprocessadores estão cada vez menores e mais eficientes em termos de consumo de energia. Esse progresso tem permitido avanços significativos nos estudos relacionados à eletromiografia, levando a uma melhor compreensão em relação aos músculos e do funcionamento da ação motora. Como resultado, as pesquisas voltadas para a inclusão de pessoas com deficiências têm se tornado mais abrangentes e acessíveis. Atualmente, no mercado já existem diversas soluções tecnológicas como próteses inteligentes gerenciadas por sinais musculares que buscam proporcionar movimentos naturais para aqueles que perderam membros, dispositivos para atletas de alto desempenho para monitorar as atividades em tempo real, permitindo um acompanhamento preciso das atividades físicas. O avanço tecnológico também tem contribuído para reduzir os custos dessas soluções, o que resulta em uma maior acessibilidade para um número cada vez maior de pessoas.

Com a ideia de abranger novos estudos na área de tecnologia e facilitar a vida das pessoas na área da saúde, este trabalho visa desenvolver um sistema supervisório para PC, que seja capaz de coletar e armazenar sinais de eletromiografia e acelerometria. Possui quatro sinais analógicos para aquisição de sinais EMG e um canal para aquisição do sinal ACC. Através do PC, é possível visualizar os sinais coletados por meio do supervisório. O sistema conta com funções gráficas em função da quantidade de amostras e da amplitude, a interface oferece quatro opções de gráficos para visualização no supervisório, proporcionando uma análise mais detalhada dos dados coletados.

1.1 Tema

Este trabalho consiste no desenvolvimento de um sistema supervisório para PC que seja capaz de armazenar e coletar dados, utilizando a plataforma BITalino para a aquisição dos sinais EMG e ACC.

1.2 Delimitação do tema

O trabalho utiliza um sistema embarcado que é o kit BITalino *MuscleBIT* que possui seis canais com alcance em quatro sinais de eletromiografia (EMG) de 10 bits, um canal para referência dos sinais EMG e um canal de acelerometria (ACC) com 6 bits, fazendo o desenvolvimento do supervisório através de um PC para visualização e análise dos dados obtidos por meio de gráficos.

1.3 Objetivos

Desenvolver um sistema em que consiga coletar e armazenar sinais de eletromiografia (EMG) e sinais de acelerometria (ACC) do sistema embarcado, e apresentar no supervisório para PC quatro opções de gráficos para visualização dos dados coletados.

1.4 Objetivos específicos

a) Analisar o funcionamento (hardware e software) de um sistema para aquisição de sinais EMG e ACC;
b) Implementar um sistema para recebimento dos dados de EMG e ACC e salvamento dos dados em um arquivo txt;
c) Desenvolver um supervisório para coletar dados dos multisinais EMG e ACC dos sensores, através do kit BITalino;
d) Implementar no supervisório opções de funções gráficas para visualização e salvamento dos dados coletados;
e) Rotular os dados adquiridos, através do pré-processamento dos sinais como filtragem, retificação e marcação do início e fim do gesto;
f) Validar o sistema através de um experimento de coleta de sinais EMG e ACC.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção são abordados conceitos relacionados e tecnologias necessárias para o desenvolvimento do trabalho. Foram consultadas fontes bibliográficas e artigos relacionados ao trabalho para ter um melhor atendimento do assunto de estudo.

Dentre os temas apresentados temos a eletromiografia, músculos esqueléticos, sinal EMG, sinal ACC, acelerometria, tipos de acelerômetros sistemas embarcados, bem como o *software* OpenSignals, sistemas supervisórios, os sistemas utilizados, hardwares que serão utilizados no desenvolvimento do projeto como os sensores ACC e EMG, plataforma BITalino, também é comentado as bibliotecas utilizadas para o trabalho na linguagem Python.

2.1 Eletromiografia

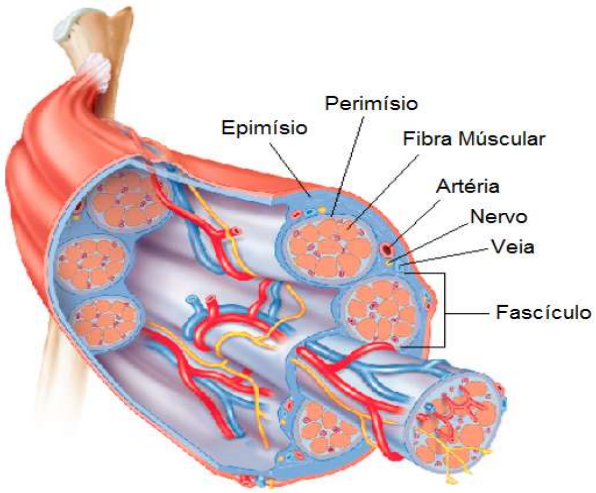
A eletromiografia é uma técnica de monitoramento para aquisição de sinais elétricos que são gerados através da contração muscular, que surge como resultado do fluxo de potenciais de ação pelas fibras musculares, dando origem ao sinal Eletromiográfico (EMG) (RICCIOTT, 2006).

Para compreender esses sinais que são enviados ao sistema muscular, é necessário primeiro estudar os músculos esqueléticos.

2.1.1 Músculos esqueléticos

Os músculos são os tecidos responsáveis pelos movimentos humanos. Através deles conseguimos manter uma posição, realizar movimentos, mover uma estrutura corporal, entre outras coisas. O músculo esquelético pode ser controlado voluntariamente e permite a produção de movimento e a estabilização das articulações, na figura 1 é possível ver a estrutura do músculo esquelético. (FREIXO, 2015)

Figura 1: Representação da anatomia do músculo esquelético.



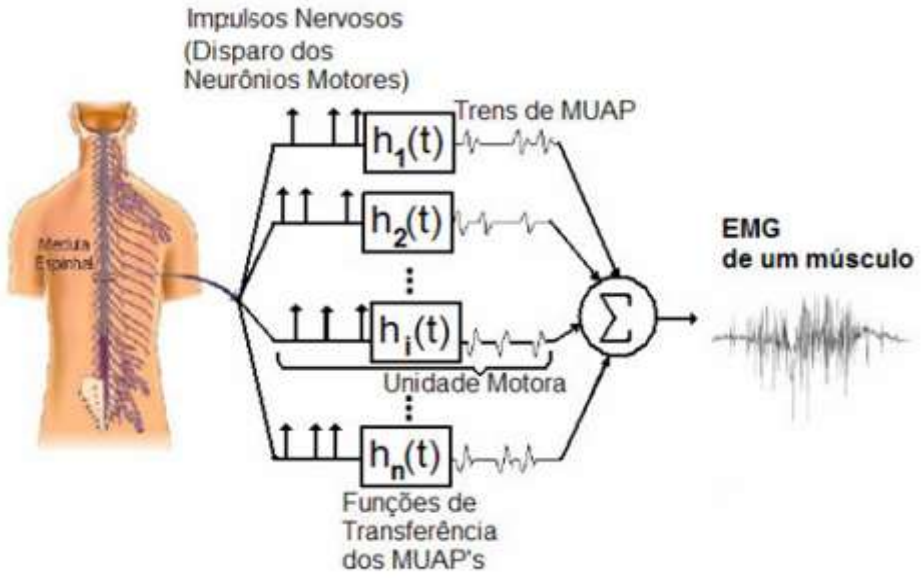
Fonte: Adaptado de (FREIXO, 2015).

O músculo esquelético geralmente tem uma espessura central que é o ventre do músculo. Cobrindo a parte externa do músculo há um tecido fibroso que é o epimísio que tem o papel de transferir a tensão muscular para o osso, o epimísio transfere várias tensões para o tendão causando a aplicação de uma força suave no osso. As fibras feixes são chamadas de fascículos e podem conter até 200 fibras musculares, e tem a função de proteger as fibras musculares e fornecer vias para os nervos e vasos sanguíneos. (FREIXO, 2015).

2.2 Sinal EMG

O sinal EMG é a soma dos diversos potenciais de ação de cada músculo, oriundos dos vários MUAPs (potencial de ação da unidade motora). Cada unidade motora é diferente entre si. Na figura 2 podemos ver a representação da geração do sinal EMG de um músculo, a partir da somatória dos trens de MUAPs (RICCIOTT, 2006).

Figura 2: Representação da geração de sinal EMG



Fonte: Adaptado de (BASMAJIAN & DeLUCA, 1985)

O sinal EMG captado na superfície da pele possui algumas características, como frequências entre 0 Hz e 1 kHz e amplitudes máximas entre 50 μV e 5 mV. (RICCIOTT, 2006).

2.3 Acelerometria

A acelerometria é um método utilizado para medir a aceleração de um sistema, normalmente utilizada para análises biomecânicas. Através do uso dos dispositivos acelerômetros é possível mensurar as acelerações do sistema a ser estudado. O dispositivo irá gerar medições em uma sequência de valores que representam o valor instantâneo da aceleração em função do tempo, ao utilizarmos o dispositivo no corpo humano, estuda-se a aceleração que o corpo gerou ou sofreu, sendo sua derivada a velocidade, posicionamento e a inclinação em relação à gravidade. (CORRÊA, 2010)

2.3.1 Acelerômetros piezoelétricos

Os sensores piezoelétricos oferecem alta sensibilidade, muito superiores, quando são comparados aos *strain-gages* e usualmente apresentam baixo custo, baixo consumo de energia e uma alta “sobrevivência a choques”, respondem a deformações menores do que 1 μm e são adequados para medição de esforços variáveis, tais como força, pressão e aceleração. Pode-se afirmar que os transdutores piezoelétricos são capazes de transformar uma pequena quantidade de energia mecânica em carga elétrica. Como os sensores piezoelétricos possuem um pequeno tamanho se tem a possibilidade de fabricação de dispositivos com sensibilidade unidirecional como o monitoramento de vibrações. (BALBINOT; BRUSAMARELLO, 2019).

2.3.2 Acelerômetros capacitivos

O acelerômetro com tecnologia capacitiva é caracterizado por apresentar uma resposta em frequência estável em função da temperatura, é um membro da família de sensores MEMS que também pode ser utilizado em sistemas estáticos, e em medições de inclinação em que a aceleração é uma constante, os acelerômetros MEMS piezorresistivos podem causar sérios problemas de estabilidade, pois são suscetíveis à contaminação na superfície. Pelos problemas de estabilidade têm sido feito esforços para desenvolver sensores MEMS, por meio de tecnologia capacitiva, que apresentem melhor estabilidade do que os piezorresistivos. (BALBINOT; BRUSAMARELLO, 2019).

Os sensores capacitivos apresentam um sinal de saída mais elevado quando comparados aos acelerômetros piezorresistivos, são inerentemente não lineares, e a medição de pequenas capacitâncias é muito difícil devido aos efeitos parasitas e a interferências eletromagnéticas do ambiente. (BALBINOT; BRUSAMARELLO, 2019).

2.3.3 Acelerômetros piezorresistivos

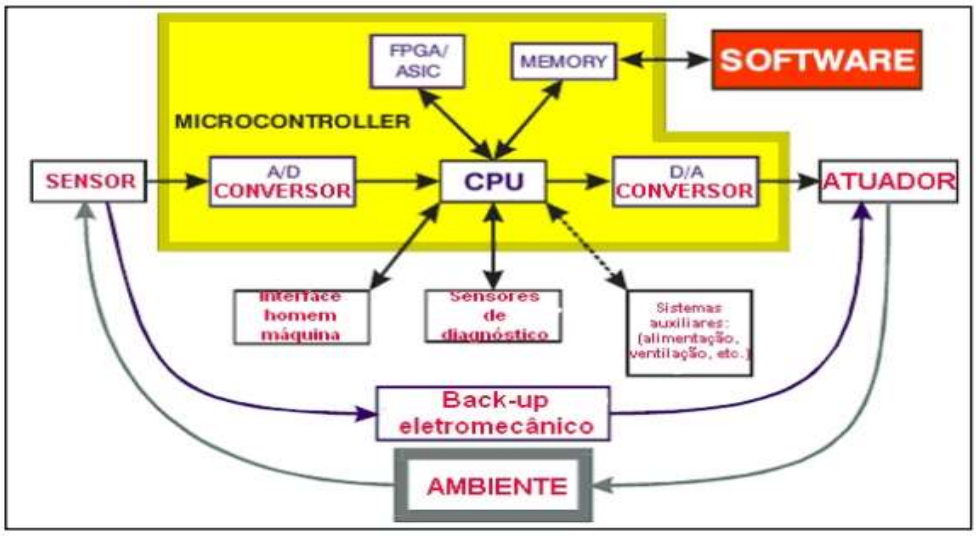
Os acelerômetros piezorresistivos são implementados com sensores strain-gages semicondutores, o que possibilita a miniaturização (sensores MEMS). São implementados com dois *strain-gages* semicondutores, sendo a configuração meia ponte de *Wheatstone* ou quatro *strain-gages* semicondutores (configuração ponte completa de *Wheatstone*). Os acelerômetros piezorresistivos podem apresentar sistemas de proteção contra sobrecarga, pois podem estar em amplitudes elevadas, o que evita danos ao sensor. Esse acelerômetro é indicado para frequências baixas, geralmente inferiores a 1 Hz, e pode ser utilizada na caracterização de sistemas estáticos. (BALBINOT; BRUSAMARELLO, 2019).

2.4 Sistemas Embarcados

O sistema embarcado, também conhecido como embutido, é um sistema computacional encapsulado ao dispositivo ou sistema que o controla, é encarregado de executar tarefas específicas predefinidas, geralmente são em sistemas de tempo real, podendo otimizar seus recursos computacionais, seu tamanho e seu custo de produção (DENARDIN; BARRIQUELO, 2019).

Para os sistemas embarcados há então a necessidade de uma inteligência que controle o funcionamento do sistema, normalmente um microprocessador ou microcontrolador é utilizado para essa função, já que ambos conseguem fazer a leitura em tempo real de sensores e enviar para os atuadores os sinais esperados (CUNHA, 2007). Na figura 3 podemos observar os componentes de um sistema embarcado, entrando no microcontrolador que tem a CPU, memória, FPGA, etc.

Figura 3: Componentes de um sistema embarcado



Fonte: Cunha (2007, p. 2).

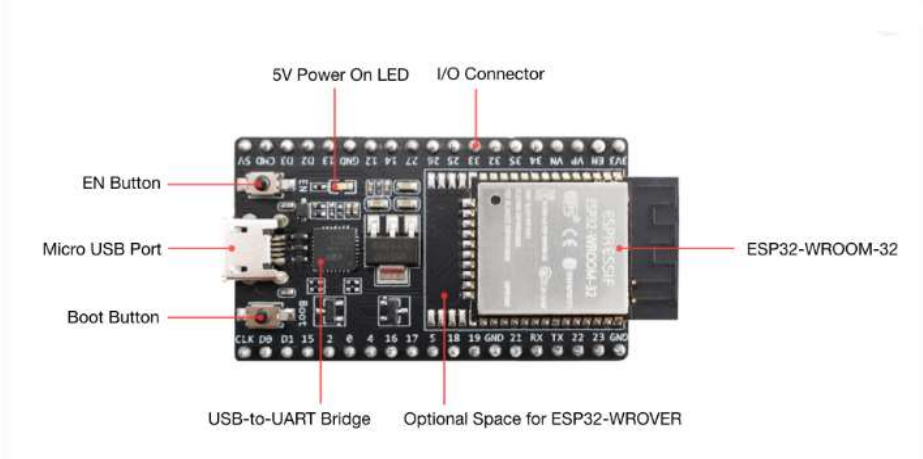
O sistema embarcado vai ser responsável pela leitura de sinais obtidos pelos sensores acelerômetros e sensores EMG para o supervisório.

2.4.1 Microcontrolador ESP32

Em 2015 a empresa chinesa *Espressif* lançou o microcontrolador ESP8266, que na época vinha com circuitos de Wi-fi embutido no chip, e a um preço acessível, em 2016 a empresa lançou seu sucessor, o microcontrolador ESP32 com mais recursos, com Wi-fi e *Bluetooth* (ESPRESSIF, 2020b).

O ESP32 é capaz de funcionar em ambientes industriais com temperatura de operação, que varia de -40ºC a +125ºC, podendo se adaptar às mudanças em condições externas. É montada em uma placa de circuito impresso (PCB) integrado com componentes como interruptores de antena embutidos, amplificador de potência, amplificador de recepção de baixo ruído, filtro e módulos de gerenciamento de energia além do microprocessador *dual-core* 32bits. O ESP32 *Devkitc* V4, visto na figura 4, pode relacionar com outros sistemas para fornecer as funcionalidades Wi-fi e *Bluetooth* através das interfaces SPI/SDIO ou I2C/UART (ESPRESSIF, 2020b).

Figura 4: ESP32-DevkitC V4 com ESP32-WROOM-32 módulo soldado

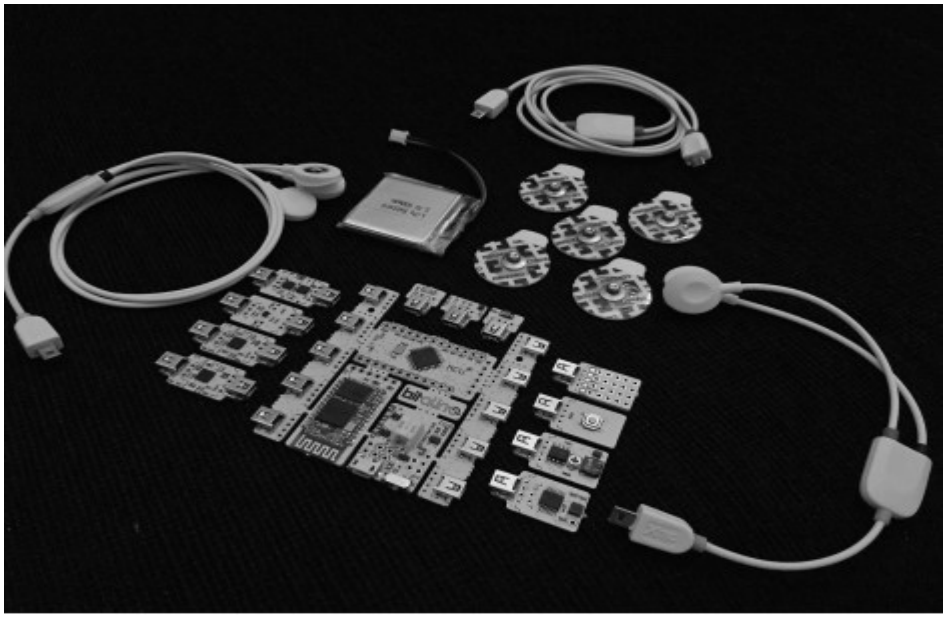


Fonte: Espressif (2016, p.1)

2.5 Plataforma BITalino

O BITalino é um kit eletrônico que disponibiliza sensores fisiológicos em diversas áreas, para aquisição em tempo real de sinais como EMG (eletromiografia), ACC, ECG, EEG, EOG, EGG e EDA, temperatura, luxímetro entre outros, podemos observar o Bitalino Revolution board kit (figura 5). Nasceu no Instituto de Telecomunicações (IT), em 2012 em parceria com a empresa tecnológica PLUX (Técnico Lisboa, 2017).

Figura 5: Bitalino (r)evolution Board kit BT (kit de desenvolvimento biomédico)



Fonte: Bitalino (2016, p.1)

2.5.1 MuscleBIT

O *MuscleBIT* (figura 6) é um kit pré-montado para aqueles que desejam medir a atividade muscular através da aquisição de sinais de eletromiografia (EMG), possui quatro sensores EMG que permitem uma medição rápida e precisa, o usuário ainda se beneficia da distância pré-fixada do eletrodo (BITALINO, 2020).

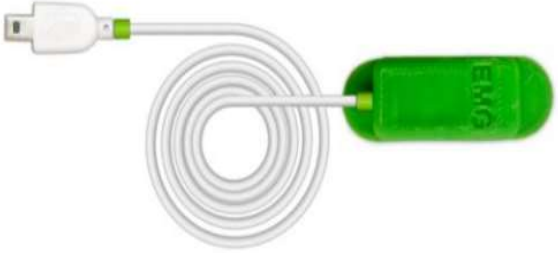
Figura 6: Kit MuscleBIT

Fonte: Bitalino PLUX (2020, p.1)

2.6 Sensores EMG

O sensor EMG (figura 7) é projetado para verificar a atividade muscular do indivíduo através da eletromiografia (EMG).

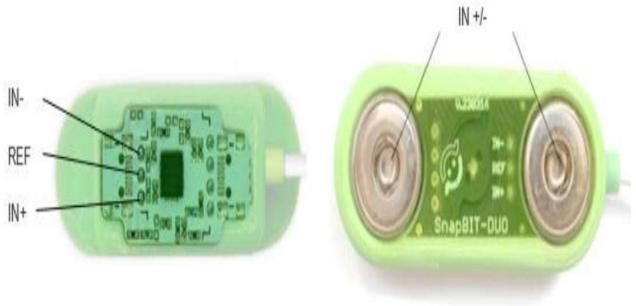
Figura 7: Sensor EMG



Fonte: Bitalino PLUX (2020, p.1)

Na figura 8 podemos ver o circuito do sensor, com duas entradas IN- e IN+ que é conectada no braço e a entrada REF (referência) que é conectada ao cotovelo. O módulo sensor EMG vai adquirir os sinais EMG conforme movimenta a mão e o braço, possui uma largura de banda de 25 Hz a 480 Hz e possui uma variação de tensão de ± 1,64 mV com um ganho de 1009 e uma rejeição ao modo comum de 86 dB.

Figura 8: Sensor EMG visão de cima (esquerda), de baixo(direita)



Fonte: Bitalino PLUX (2020, p.1)

Na figura 9 conforme o *datasheet* do sensor podemos observar a posição dos eletrodos no braço da pessoa, a qual são as entradas IN- e IN+ e a entrada de referência.

Figura 9: Posição eletrodos

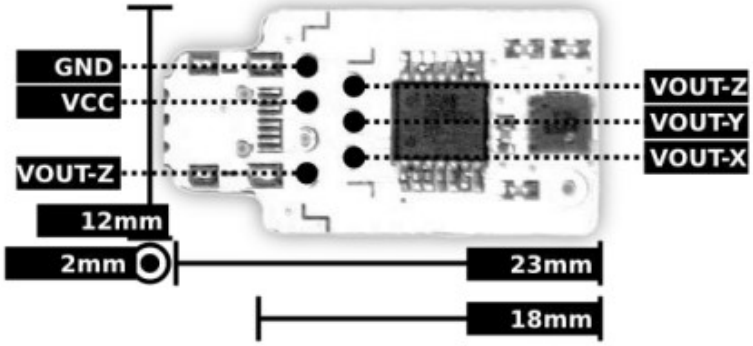
Fonte: Bitalino PLUX (2020, p.1)

2.6.1 Sensores ACC (acelerômetro)

O sensor ACC pode adquirir dados de eventos cinemáticos e biomecânicos, pois o movimento produz uma aceleração que será convertida em um valor numérico e lida pelo sensor (BITALINO, 2020). As aplicações desse sensor podem ser detecção de postura, estimativa de amplitude de movimento e detecção de queda, entre outras

aplicações possíveis que o acelerômetro (ACC) consegue fazer. Na figura 10 podemos ver os pinos do sensor que possui três eixos (VOUT-Z, VOUT-Y, VOUT-X). Geralmente apenas o eixo Z é conectado (BITALINO, 2020).

Figura 10: Pinos do acelerômetro (ACC)



Fonte: Bitalino PLUX (2020, p.1)

2.7 Comunicação sem fio

A comunicação sem fio desempenha um papel fundamental em nossa sociedade atual, permitindo a conectividade e troca de dados entre dispositivos em várias aplicações. Nesta seção, serão abordadas algumas das principais tecnologias de comunicação sem fio, como o *Bluetooth* e o Wi-fi.

2.7.1 Comunicação Bluetooth

Bluetooth é um padrão de comunicação sem fio de curto alcance, baixo custo e baixo consumo de energia, *Bluetooth* tem se tornado amplamente utilizado em diversos dispositivos e já representa uma parcela significativa do mercado *wireless*. Dentre os dispositivos que utilizam *Bluetooth* podemos incluir os dispositivos inteligentes, como celulares, PC, lâmpada, periféricos, dispositivos embarcados, entre outros. (SIQUEIRA, 2006).

A tecnologia *Wireless Personal Area Network* (WPAN), baseada na especificação *Bluetooth*, é agora um padrão IEEE sob a denominação 802.15.1 WPANs. A arquitetura *Bluetooth* consiste em dois componentes, um *transceiver* (hardware) e uma pilha de protocolos (software), esses componentes tornam possível a conexão dos dispositivos e a troca de dados entre os mesmos. (SIQUEIRA, 2006).

2.7.2 Comunicação Wi-fi

Foi em 1990 que o IEEE (*Institute of Electrical and Electronics Engineers*) estabeleceu um comitê para definir uma norma para as redes locais sem fios. A primeira versão desta norma foi aprovada em 1997 sob o nome de IEEE 802.11, e suportava taxas de transmissão de 1 e 2 Mbps. Com o tempo aprovaram outras normas e os dispositivos que implementavam a norma IEEE 802.11b, acabaram por conquistar uma importante quota do mercado devido ao seu baixo custo e por serem disponibilizados para venda antes dos dispositivos que implementam a norma IEEE 802.11a. O Wi-fi que é a abreviação para *"Wirelles Fidelity*" que significa fidelidade sem fio é a tecnologia que não faz uso de cabos e é transmitida através de frequências de rádio. (COSTA, 2013).

Em 2003, o IEEE aprovou a norma 802.11g que opera na gama de frequência de 2.4 GHz e consegue atingir taxas de transmissão até 54 Mbps e em 2012 foi aprovado a norma revisada, com as correções e incorporou várias emendas em relação às redes sem fios. (COSTA, 2013)

2.8 Sistemas Supervisórios

Nesta seção serão analisados alguns sistemas supervisórios, que são softwares para monitoramento e visualização de dados de um sistema, como o framework QT.

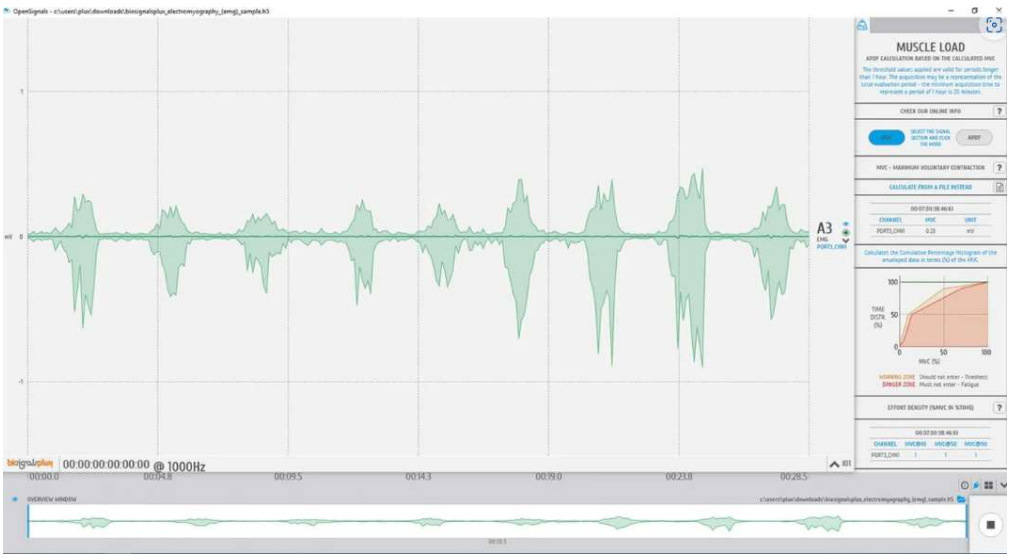
2.8.1 QT Designer

O software QT é um *framework* que interage em diferentes classes de programas, como a interface gráfica e o ambiente de desenvolvimento, foi desenvolvida pela *Trolltech* e é um software *open source*. Pode ser usado nos sistemas operacionais Linux e Windows para o desenvolvimento de sistemas supervisórios e utiliza a linguagem C++ e pode ser usado tanto em *desktop* como em plataformas incorporadas (QT, 2022a).

2.8.2 OpenSignals

OpenSignals é um conjunto de software versátil para visualização de biosinais em tempo real, foi desenvolvido pela PLUX e é compatível com todos os dispositivos PLUX. As principais funcionalidades incluem aquisição de dados de sensores de vários canais e dispositivos simultaneamente, visualização e gravação de dados. Possui um conjunto de complementos de análise de dados para criar relatórios a partir dos dados registrados e extrair recursos diretamente dos sinais sem precisar fazer nenhuma codificação. Na figura 11 podemos ver a utilização do OpenSignals para obtenção de sinais EMG utilizando biosignalplux e sensores BITalino EMG. (PLUX, 2022a).

Figura 11: OpenSignal Visualização de sinais EMG



Fonte: (PLUX, 2019)

2.9 Linguagem de programação Python

Python é uma linguagem de programação, que foi projetada, para cientistas, engenheiros e analistas de dados, programadores, etc. A linguagem apresenta funcionalidades como edição, análise, depuração, geração de gráficos, inspeção, entre outras funcionalidades, além de oferecer integração com pacotes científicos conhecidos como NumPy, SciPy, Pandas, IPython, QtConsole, Matplotlib, SymPy, Jupyther Notebook entre outros recursos que possam integrar. Na figura 12 podemos observar as funcionalidades do Python.

Figura 12: Diversas funcionalidades do Python

Fonte: Anaconda Distribution

2.9.1 Bibliotecas utilizadas

Para o desenvolvimento do supervisório para PC em Python é necessário instalar bibliotecas que permitem a troca ou obtenção de informações, para, por exemplo, podermos conectar o *Bluetooth* do computador ao do BITalino e conseguirmos gerar gráficos, conectar o Python ao software de interface gráfico com o Qt Designer.

2.9.2 Numpy

NumPy é uma biblioteca para a linguagem de programação Python, é desenvolvido e mantido no *GitHub* pela comunidade, NumPy suporta o processamento de grandes *arrays* dimensionais, juntamente com uma grande coleção de funções matemáticas de alto nível, como funções geradoras de números aleatórios, rotinas de álgebra linear, entre outras funções e também oferece suporte a uma variedade de plataformas de hardware e computação. Para a instalação no Python é necessário digitar no prompt de comando da Anaconda *pip install numpy* (PYTHON SOFTWARE FOUNDATION, 2023).

2.9.3 Pyserial

Esse módulo encapsula o acesso para a porta serial, além de fornecer *back-ends* para Python em execução no Windows, para a instalação no Spyder Python é necessário entrar no prompt de comando da Anaconda e digitar *pip install pyserial* (PYTHON SOFTWARE FOUNDATION, 2023).

2.9.4 PyBluez

O módulo PyBluez permite que o código Python acesse os recursos *Bluetooth* da máquina host. Sendo necessário no Windows baixar do *github* o módulo e colocar na pasta da Anaconda em site *packages* ou através da instalação *pip*, com isso conseguimos ter acesso ao *Bluetooth* do computador (PYTHON SOFTWARE FOUNDATION, 2023).

2.9.5 Matplotlib

É uma biblioteca para criar visualizações de gráficos, seja animada, interativas, estáticas em linguagem Python, podemos visualizar as aquisições de sinais pelos sensores EMG e ACC através de gráficos, podendo modificar os parâmetros de como serão visualizados no supervisório (PYTHON SOFTWARE FOUNDATION, 2023).

2.9.6 PyQT5

O Qt é um conjunto de bibliotecas C++ de plataforma cruzada que implementam APIs de alto nível para acessar muitos aspectos de sistemas modernos de desktop, como o desenvolvimento de interfaces gráficas. PyQt5 é um conjunto abrangente de ligações Python para Qt v5. Ele é implementado com mais de 35 módulos de extensão e permite que o Python seja usado como uma linguagem de desenvolvimento de aplicativos alternativa ao C++ em todas as plataformas suportadas. (Python Package Index, 2023).

2.9.7 Pyqtgraph

PyQtgraph é uma biblioteca de gráficos puro-python construída em Pyqt5, Pyside2 e numpy e destina-se ao uso em aplicações matemáticas de engenharia. A biblioteca é muito rápida devido a sua forte influência de numpy para processamento de números, foi utilizado para a ferramenta Qt *GraphicsView* para exibição de gráficos 2D e OpenGL para exibição em 3D. (Python Package Index, 2023).

Nesse capítulo foram abordados conceitos que serão utilizados no desenvolvimento do supervisório para PC, estudamos assuntos como a eletromiografia, sensores de eletromiografia (EMG), músculos esqueléticos para compreender como funciona a aquisição desses sinais. Sinal EMG, sinal ACC, acelerômetros utilizados na indústria como piezoelétrico, capacitivo e piezorresistivos, sistemas embarcados, ESP32. Também foi abordado a plataforma BITalino, utilizada para a aquisição dos sinais através dos eletrodos, foi pesquisado sobre os sistemas supervisórios como o QT, que será a base para entender o funcionamento dos mesmo para o desenvolvimento do supervisório para PC através da linguagem Python e a comunicação que será utilizada no desenvolvimento do supervisório como *Wifi* e *bluetooth*.

3 TRABALHOS RELACIONADOS

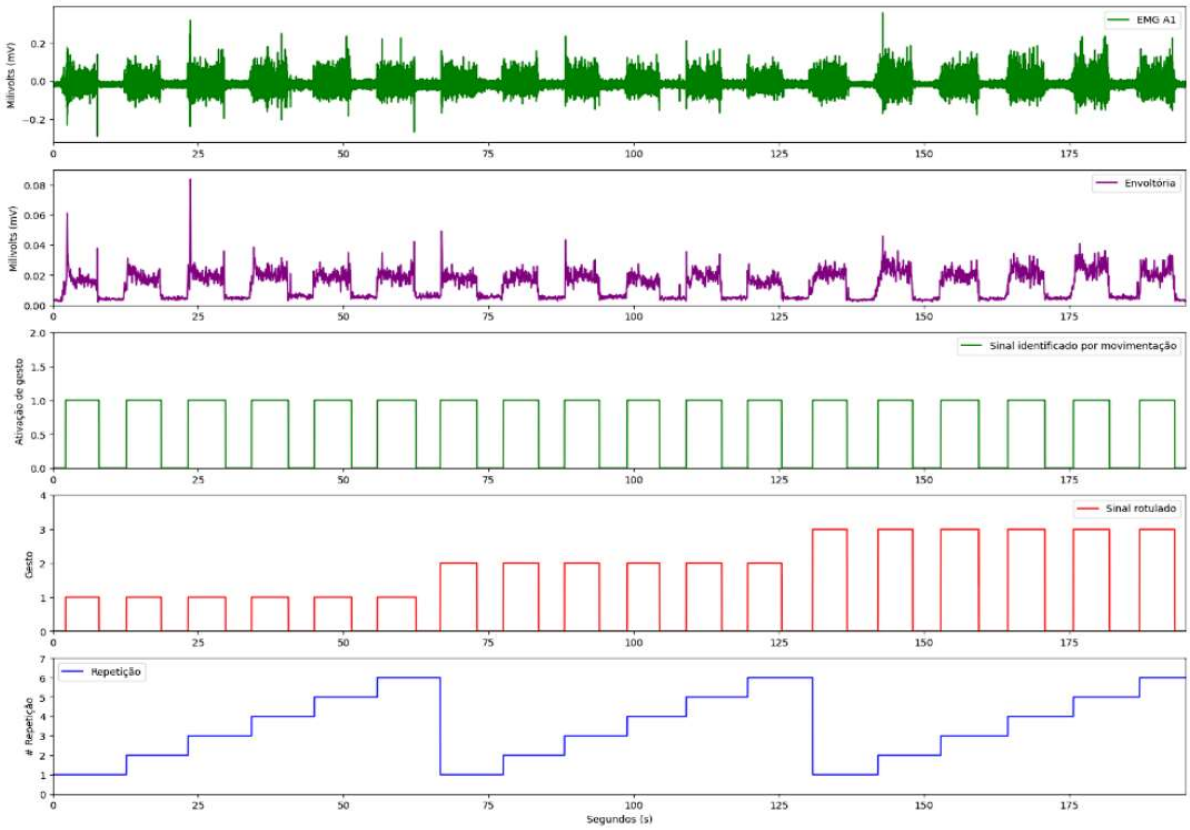
Esta seção apresenta trabalhos relacionados com o trabalho proposto. Os mesmos possuem problemas similares e contribuem diretamente com a solução proposta.

3.1 Desenvolvimento de banco de dados de movimentos/gestos de mão através de sinais de eletromiografia

Trabalho desenvolvido por Beloni (2022) que fez um sistema capaz de armazenar sinais EMG, extrair seus dados e registrá-los em um banco de dados, permitindo a visualização dos sinais obtidos, para reconhecimento de gestos de mão e rotulagem. O trabalho foi desenvolvido a fim de armazenar em formato semelhante ao padrão de banco de dados Ninapro, mas com uma taxa de atualização do sinal em 1 kHz. O algoritmo desenvolvido foi baseado no projeto de detecção de movimentos disponibilizado pela Biosignalplux, foi feito diversos processos como um filtro passa-banda Butterworth para atuar no sinal EMG e também processo de retificação e suavização.

O resultado também integrou os parâmetros de aquisição do sinal eletromiográfico no início do arquivo, respeitando os dados mostrados em função do tempo, rotulados por tipo de movimento e sua repetição. Na figura 13 podemos ver a visualização de todos os sinais que o trabalho obteve, exibindo o sinal de entrada EMG do canal 1 (em cor verde), em seus valores em milivolts (mV), a envoltória do sinal de entrada, aparecendo em cor violeta, no terceiro gráfico, mostrou o sinal agrupado, exibindo o seu início e fim de atividade muscular, mostrou o sinal rotulado por exercício (0: Repouso, 1: Polegar para cima, 2: Extensão do dedo indicador e médio, 3: Punho fechado) em vermelho, por último foi plotado o gráfico de repetição do sinal (em azul).

Figura 13: Visualização de todos os sinais obtidos



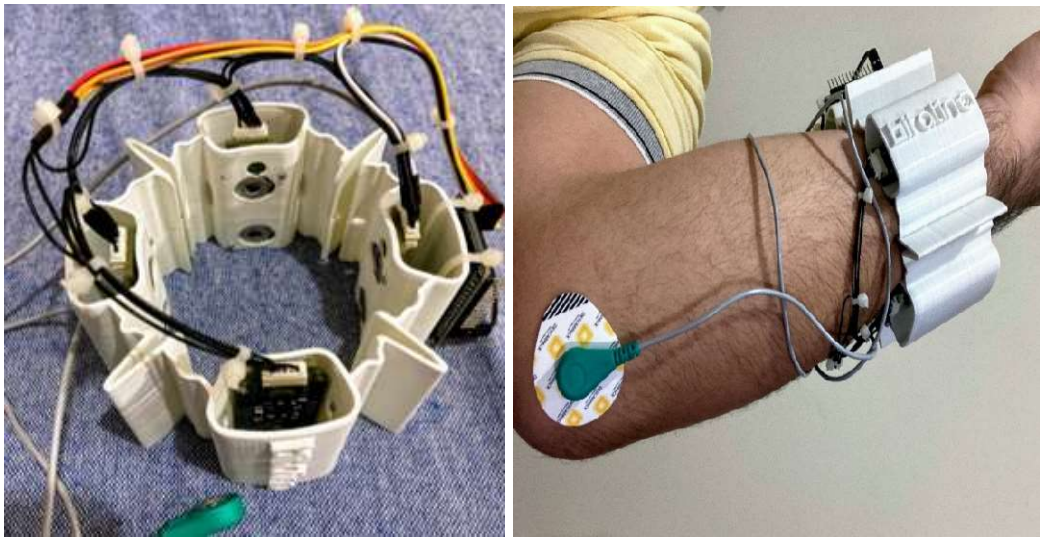
Fonte: (BELONI, 2022)

3.2 Projeto e desenvolvimento de um protótipo para aquisição de multisinais EMG

Trabalho desenvolvido por Santos (2021) que fez um protótipo para aquisição de multisinais EMG, utilizando um bracelete para até quatro sinais EMG, através de uma ESP32 e comunicação *Bluetooth* com o auxílio do software OpenSignal, foram salvos os dados para análises futuras. Para a aquisição dos sinais foram utilizados os quatro eletrodos que estão conectados na ESP32 com *firmware* customizado de BITalino e a partir disso foram coletados os dados através do software OpenSignals.

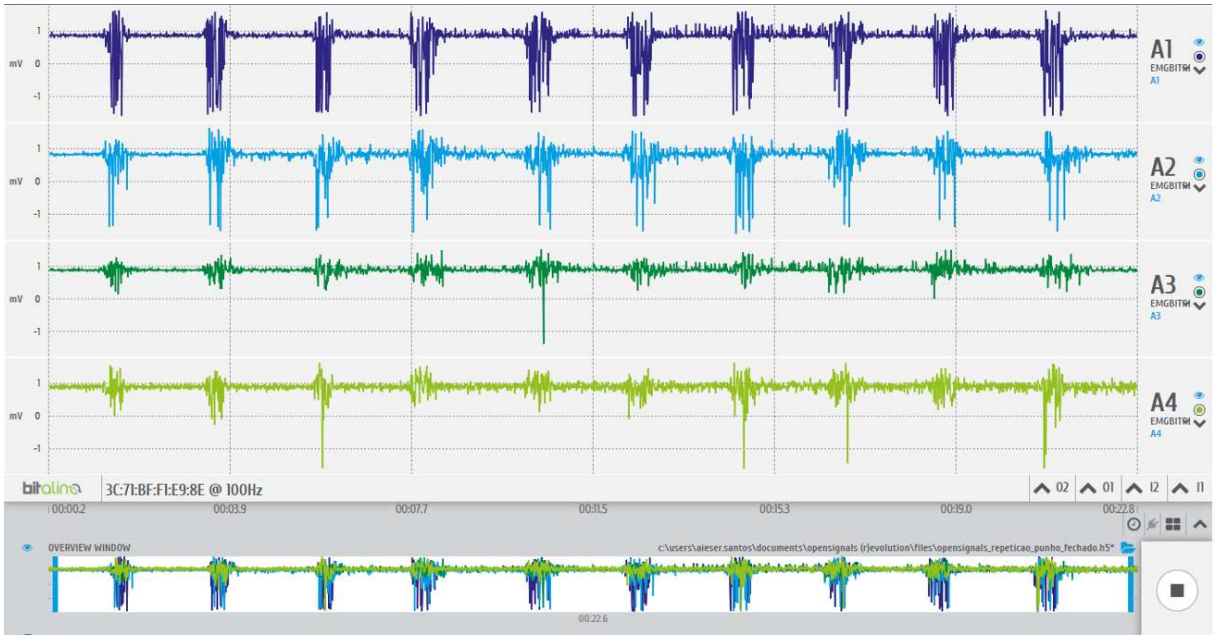
O circuito projetado foi capaz de condicionar os sinais de forma adequada, filtrando os principais ruídos e amplificando os sinais EMG para os ADCs do ESP32. Na figura 14 e 15 podemos ver respectivamente o protótipo do bracelete montado e a geração de sinais EMG com dez movimentos de polegar para cima.

Figura 14: Protótipo bracelete para sinais EMG



Fonte: (SANTOS, 2021)

Figura 15: Geração de sinais EMG com dez movimentos de polegar para cima



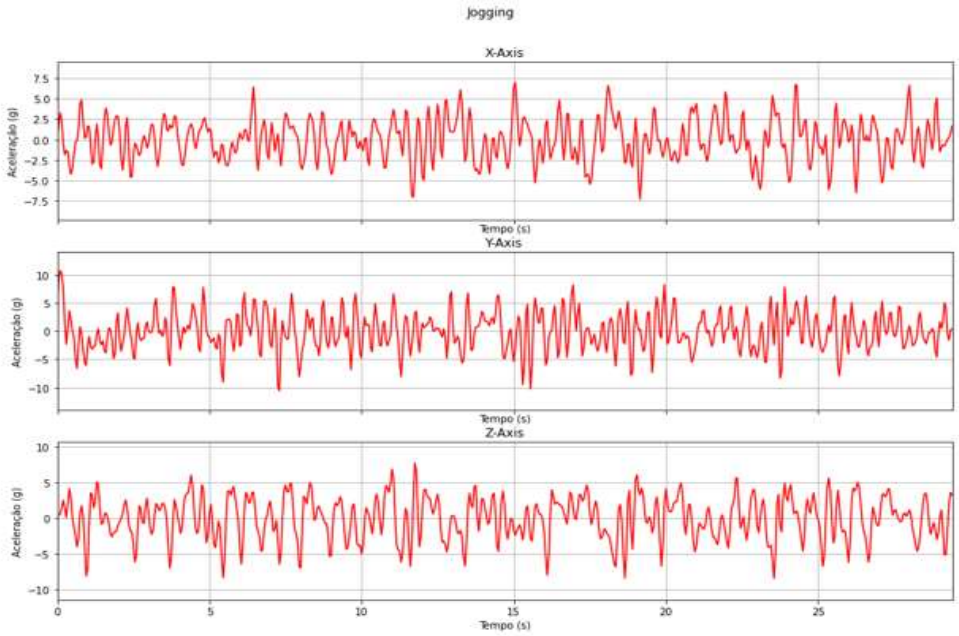
Fonte: (SANTOS, 2021)

3.3 Reconhecimento dos movimentos humanos utilizando inteligência computacional

Trabalho desenvolvido por Pinto (2022) que desenvolveu um algoritmo que se utiliza da inteligência artificial para classificar atividades diárias do corpo humano, como andar, ficar parado, correr, etc. Foi utilizado um banco de dados já existente que

foi captado por um sensor acelerômetro, e por fim foram classificados por uma rede neural MLP. O banco de dados escolhido foi o banco desenvolvido pela WISDM *Labs* durante uma pesquisa em 2010. O banco foi utilizado em um artigo, que serviu como comparativo com o classificador desenvolvido no trabalho. As *features* definidas foram 6 tipos base, que em alguns casos foram desdobrados para cada eixo, assim totalizando 43 *features* a serem utilizadas para classificar os dados. Na figura 16 podemos ver um exemplo dos eixos obtidos para as atividades que foram realizadas.

Figura 16: Eixos X, Y e Z para o primeiro trecho da atividade 1



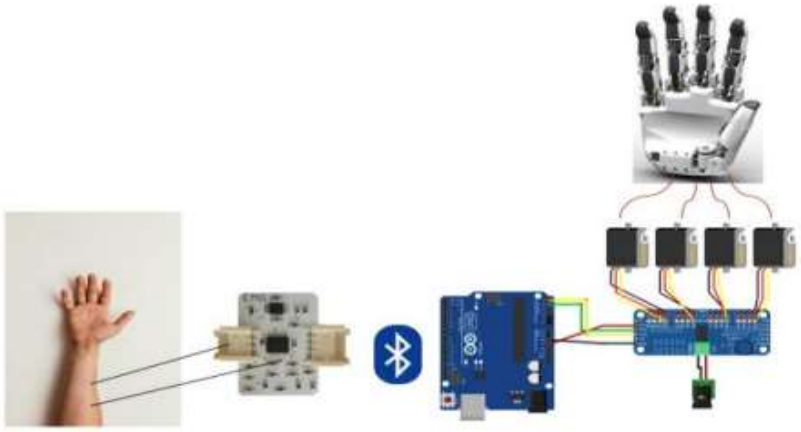
Fonte: (PINTO, 2022)

Os objetivos do trabalho foram alcançados, uma vez que foi selecionado um banco existente, houve o desenvolvimento das rotinas para o reconhecimento dos movimentos e a definição de intervalos de janelamento. Um dos principais pontos a serem destacados no trabalho, foi a dificuldade da identificação de movimentos relacionados a subida e descida de escadas. O movimento de descida de escadas não pode ser identificado com o classificador desenvolvido, sendo na maioria dos testes confundido com o movimento de subida. Quando os dois movimentos foram unidos em apenas uma classe, percebeu-se que ainda assim, existe uma certa dificuldade na identificação destes movimentos.

3.4 Análise comparativa dos resultados de classificação de sinal EMG com base em algoritmo de aprendizado de máquina

Neste artigo publicado por Turgunov et al. (2021) foi feito uma análise de classificação de sinal EMG com base em algoritmo de aprendizado de máquina, foi adquirido informações sobre a estrutura, módulos e características do hardware e *software* complexo para classificar os movimentos das mãos. Foi desenvolvido utilizando o dispositivo BITalino, e tinha como objetivo principal classificar mais movimentos de mãos usando menos sensores, para que no futuro seja possível produzir mais fácil e barato as miopróteses. Na figura 17 podemos observar a estrutura geral do hardware utilizado a plataforma BITalino e Arduino para a aquisição de sinais EMG e análise dos resultados da classificação.

Figura 17: Estrutura do hardware e software para reconhecimento dos movimentos das mãos com base no sinal EMG.

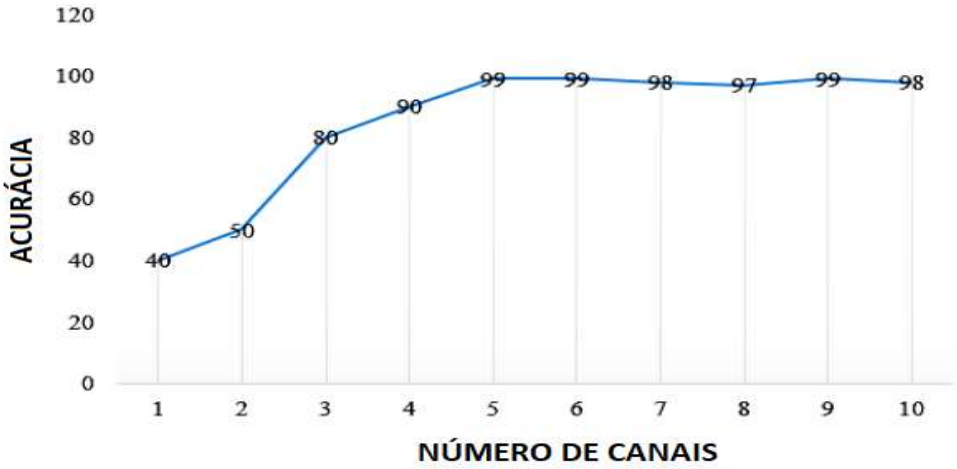


Fonte:(TURGUNOV ET AL, 2021)

Neste estudo foi determinado a relação entre o número de sensores e os movimentos das mãos, isso indicou a necessidade de um número máximo de sensores correspondentes ao número de movimentos das mãos. A precisão do número de canais e a complexidade do processo de cálculo são diretamente proporcionais. Portanto, a escolha correta do número correto de canais eliminará estes problemas.

Como os experimentos foram realizados principalmente em um dispositivo de canal único, a influência do número de movimentos na precisão da classificação foi observada. Foi possível classificar com precisão necessária até 3 ou 4 movimentos em um dispositivo de canal único. Considerando o grande fluxo de dados em um dispositivo multicanal, o número de sensores é de grande importância, afetando a precisão da classificação. Quanto mais sensores, mais difícil será calcular os parâmetros do sinal em tempo real e menor o grau de precisão da classificação. Assim, chegamos à conclusão de experimentos em um dispositivo multicanal que é possível executar ações multifuncionais usando um máximo de 5 sensores (número de canais). Na figura 18 podemos observar essa quantidade de canais e a precisão do resultado do artigo.

Figura 18: Número de canais



Fonte:(TURGUNOV ET AL, 2021)

De acordo com os resultados, o objetivo do trabalho foi alcançado com o número de sensores para uma classificação com uma alta precisão dos movimentos de mãos através da aquisição de sinal EMG.

Os trabalhos relacionados apresentados nesta seção ofereceram contribuições relevantes para o desenvolvimento do trabalho proposto, os trabalhos abordam problemas similares, fornecendo soluções para lidar com questões relacionadas ao processamento e reconhecimento dos sinais de eletromiografia e acelerometria. Destaca-se no trabalho relacionado, desenvolvido por Beloni (2022) a criação do

banco de dados de movimentos/gestos de mão utilizando sinais de eletromiografia, o uso de técnicas de pré-processamento como filtragem, retificação e marcação do início e fim do gesto, contribuindo para a melhoria dos sinais obtidos. O projeto realizado por Santos (2021) foi focado no desenvolvimento de um protótipo para aquisição de multisinais EMG. O bracelete projetado permitiu a coleta de até quatro sinais EMG por meio de uma ESP32 e comunicação *Bluetooth*. Destaca-se o adequado condicionamento dos sinais e a utilização de técnicas de filtragem para minimizar ruídos e amplificar os sinais EMG, possibilitando uma aquisição precisa dos dados.

Já o trabalho de Pinto (2022) se concentrou no reconhecimento de movimentos humanos utilizando inteligência computacional. A utilização de um banco de dados existente, juntamente com a implementação de um algoritmo baseado em uma rede neural MLP, permitiu a classificação de atividades diárias do corpo humano. Por fim, o artigo de Turgunov et al. (2021) apresentou uma análise comparativa dos resultados de classificação de sinal EMG utilizando algoritmos de aprendizado de máquina. A estrutura de hardware e software desenvolvida, juntamente com a utilização do dispositivo BITalino, permitiu a classificação de movimentos das mãos com base nos sinais EMG adquiridos. Destaca-se a importância da escolha correta do número de canais para obter uma precisão satisfatória na classificação.

4 METODOLOGIA

Nesta etapa do trabalho, são detalhados os procedimentos e arquitetura do sistema supervisório para PC, como a aquisição dos sinais EMG e do sinal ACC através da linguagem Python, a montagem da interface gráfico do supervisório no software Qt Designer, e as análises dos dados obtidos através dos gráficos.

4.1 Arquitetura do sistema

O projeto consiste no desenvolvimento do supervisório para PC, o kit utilizado foi o BITalino *MuscleBIT* que possui 6 canais disponíveis, sendo 4 canais de 10 bits para os sinais EMG e 1 canal de 6 bits para o sinal ACC. O objetivo do projeto foi de coletar sinais EMG e ACC do usuário, o sensor EMG é conectado aos canais da plataforma BITalino e fixado na pessoa por meio de eletrodos ao redor do braço e através da movimentação dos músculos é possível obter os gestos para a coleta do sinal EMG. Já para o sinal ACC, movimentamos o sensor para conseguir a aquisição do movimento. A comunicação do sistema ocorre via *Bluetooth* e os dados são coletados pelos sensores e enviados para o BITalino, que realiza a leitura no programa desenvolvido em Python. Os gráficos apresentados na coluna do sinal EMG são de 10 bits, variando de 0 a 1023, enquanto o sinal ACC é representado por 6 bits, variando de 0 a 63. Na figura 19 é possível observar o diagrama da arquitetura do sistema implementado.

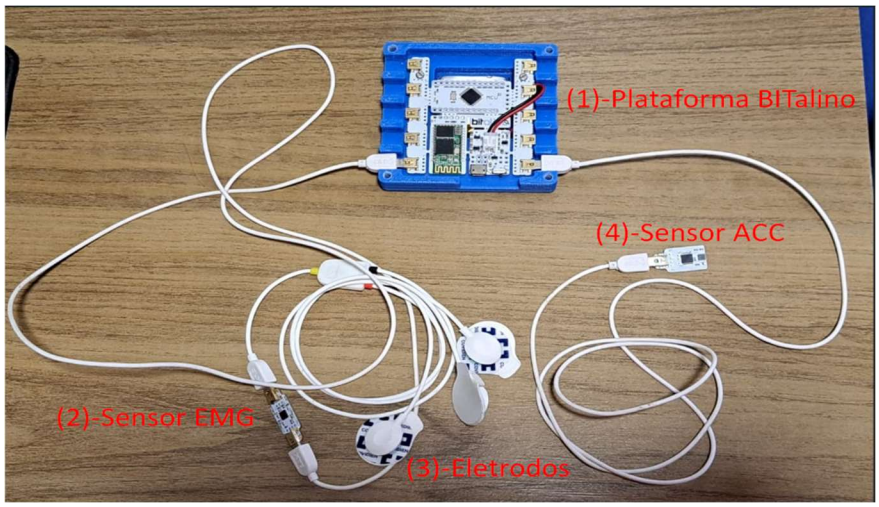
Figura 19: Diagrama da Arquitetura do Sistema

Fonte: Elaborado pelo Autor.

O BITalino envia os dados para o supervisório desenvolvido para PC, transmitindo valores predefinidos, assim simulando os sinais EMG e ACC nas entradas analógicas. Nos gráficos do supervisório, é possível visualizar a dashboard do sistema, bem como selecionar os gráficos desejados. Existem quatro opções disponíveis: opção 1 com 1 canal EMG e 1 canal ACC, opção 2 com 2 canais EMG e 1 canal ACC, opção 3 com 3 canais EMG e 1 canal ACC e opção 4 com 4 canais EMG e 1 canal ACC; a taxa de amostragem utilizada é de 1 kHz.

Conforme a figura 20, para adquirir os sinais EMG, são utilizados eletrodos nos membros superiores, os eletrodos são conectados por cabos no sensor EMG, que por sua vez é conectado na plataforma BITalino. Da mesma forma o sensor ACC é conectado na plataforma BITalino, o sensor ACC não requer o uso de eletrodos, e pode ser colocado no corpo do usuário para obter os movimentos ou deixado em repouso.

Figura 20: 1- Plataforma BITalino; 2- Sensor EMG; 3- eletrodos; 4- Sensor ACC



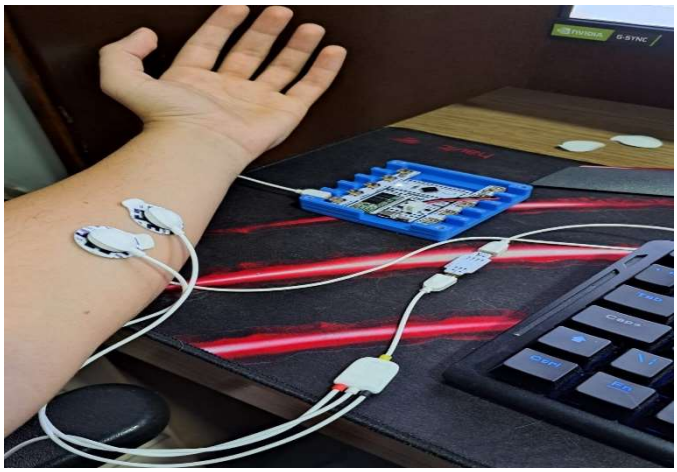
Fonte: Elaborado pelo Autor.

4.2 Aquisição dos Sinais

Como primeira implementação e experimento, foi realizada a aquisição de apenas 1 canal EMG e 1 canal ACC, para melhor visualização. Os sensores foram conectados à plataforma BITalino, com o sensor EMG no canal A1 e o sensor ACC no canal A5. Para a aquisição do sinal EMG, utilizou-se o sensor EMG conectado na

plataforma BITalino com os eletrodos posicionados conforme ilustrado na figura 21. Ao todo são posicionados três eletrodos, utilizando a configuração bipolar. Os eletrodos de entrada (vermelho e preto) são posicionados no antebraço, enquanto a entrada de referência é conectada no cotovelo. Preferencialmente o usuário deve realizar os ensaios sentado, com o braço apoiado em uma superfície plana que permita movimentos fáceis.

Figura 21: Posição dos Eletrodos com a plataforma BITalino

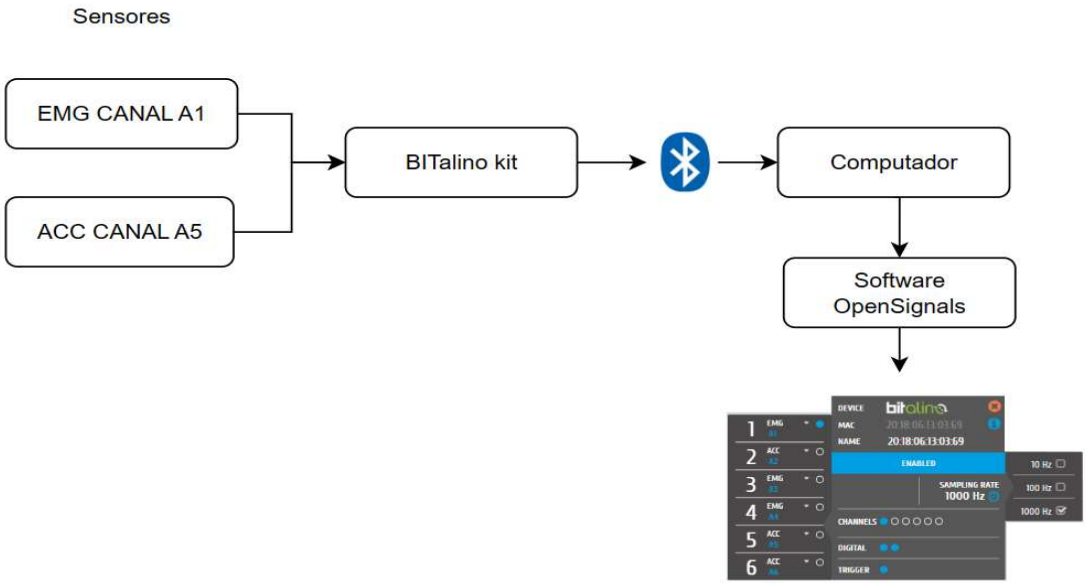


Fonte: Elaborado pelo Autor.

4.2.1 Conexão com BITalino

Para os testes iniciais com o BITalino, o software OpenSignals foi utilizado e configurado no computador com o PIN de 1234 e conectado via *Bluetooth*. Em seguida, foi estabelecida a conexão com o software, conforme mostrado na figura 22. No projeto é adotado a frequência de 1 kHz e o canal A1 para o sinal EMG, como estamos usando apenas um sensor, o BITalino automaticamente atribui para uma resolução de 10 bits.

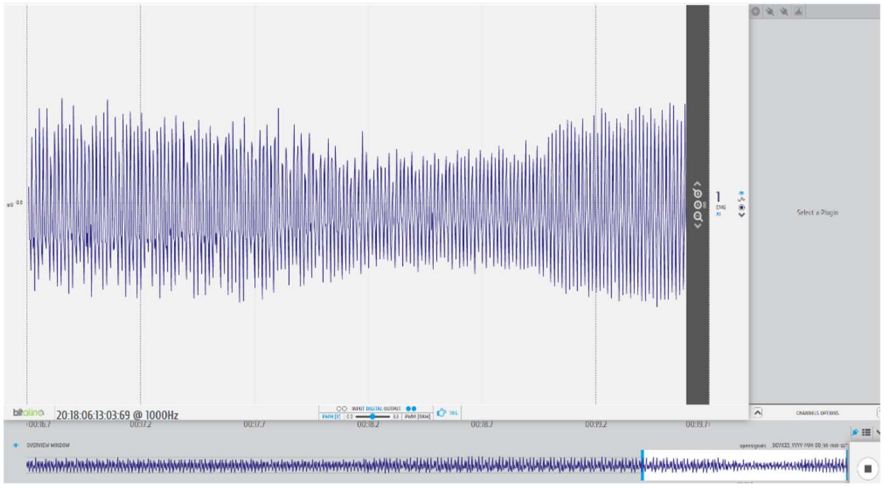
Figura 22: Fluxograma conexão BITalino



Fonte: Elaborado pelo Autor.

Com esses passos é possível obter o sinal EMG em real-time, que varia conforme o movimento do braço. O sinal pode ser visualizado no software OpenSignals, conforme ilustrado na figura 23, o sinal aumenta assim que a mão é movimentada, seja com a mão fechada, mão esticada, etc.

Figura 23: Aquisição Sinal EMG - OpenSignals



Fonte: Elaborado pelo Autor.

4.2.2 Aquisição de inicial de sinais e geração do banco de dados no OpenSignals

Realizamos os testes iniciais utilizando o Software OpenSignals, para verificar o correto funcionamento do sensor por meio do Python. Para essa etapa, utilizamos o sensor acelerômetro no canal A5 do BITalino, pois o mesmo não requer a conexão dos eletrodos, o que facilita a obtenção dos dados. O sensor foi colocado em repouso sobre uma superfície, e a taxa de amostragem utilizada é de 1 kHz, iniciamos a coleta por meio da aquisição em tempo real do software.

Após um curto intervalo de 10 segundos, salvamos um arquivo de texto para registrar os valores obtidos pelo sensor. Conforme figura 24 podemos observar que o software OpenSignals nos fornece 6 colunas para visualização, a primeira coluna representa a sequência dos números que vai de 0 a 15, as colunas seguintes “I1” e “I2”, são referentes às entradas digitais, enquanto as colunas “O1” e “O2” são referentes às saídas digitais. Pôr fim, a última coluna “A5” nos fornece os valores obtido pelo sensor naquele instante de tempo. Observamos que o valor do ACC está em um valor em torno de 600. De acordo com o *datasheet*, quando temos até 5 canais conectados, a resolução é de 10 bits, independentemente do canal utilizado. No caso de 5 ou mais canais, os últimos canais têm uma resolução de 6 bits, o software gera um arquivo de texto com milhares de linhas para posterior análise.

Figura 24: Visualização TXT do software OpenSignals

```
# OpenSignals Text File Format. Version 1
# {"20:18:06:13:03:69": {"position": 0, "device": "bitalino_rev", "device name": "20:18:06:13:03:69", "sampling
rate": 1000, "resolution": [4, 1, 1, 1, 1, 10], "firmware version": 1281, "comments": "", "keywords": "", "mode":
2, "sync interval": 2, "date": "2023-4-2", "time": "18:25:58.313", "channels": [5], "sensor": ["ACCBITREV"],
"label": ["A5"], "column": ["nSeq", "I1", "I2", "O1", "O2", "A5"], "special": [{}], "digital IO": [0, 0, 1, 1]}}
# EndOfHeader
0       0       0       1       1       610
1       0       0       1       1       627
2       0       0       1       1       619
3       0       0       1       1       616
4       0       0       1       1       627
5       0       0       1       1       611
6       0       0       1       1       620
7       0       0       1       1       623
8       0       0       1       1       611
9       0       0       1       1       627
10      0       0       1       1       615
11      0       0       1       1       615
12      0       0       1       1       627
13      0       0       1       1       609
14      0       0       1       1       620
15      0       0       1       1       623
0       0       0       1       1       612
1       0       0       1       1       627
```

Fonte: Elaborado pelo Autor.

4.3 Conexão BITalino com Python

Para realizar a conexão com o BITalino precisamos da API (*Application Programming Interface*) da BITalino Revolution, que nos fornece as ferramentas necessárias para interagir com o BITalino usando Python. Do mesmo modo que as outras APIs, ela é instalada no prompt de comando da Anaconda, por meio do comando "*pip install bitalino*" (PYTHON PACKAGE INDEX, 2023).

4.3.1 Endereço MAC da BITalino

Para estabelecer a conexão *Bluetooth* com o BITalino em Python, precisamos obter o endereço MAC do dispositivo ao qual queremos conectar. Podemos encontrar essa informação na parte de trás do dispositivo, como mostrado na figura 25.

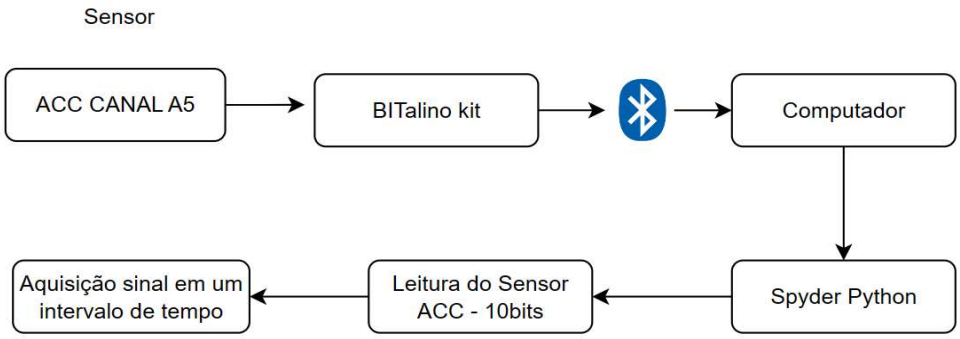
Figura 25: Endereço MAC do BITalino

Fonte: Elaborado pelo Autor.

4.4 Primeira versão do supervisório para PC

No desenvolvimento inicial do supervisório que substituirá o software OpenSignals, é realizado a leitura do sensor ACC através da linguagem de programação Python, o sensor ACC é conectado no canal A5 do BITalino e utilizando uma taxa de 1 kHz é obtido a aquisição do sinal. Na figura 26, podemos observar o diagrama da versão inicial do supervisório.

Figura 26: Fluxograma aquisição sinal Python



Fonte: Elaborado pelo Autor.

Na primeira versão do supervisório, é utilizado apenas um canal analógico para a aquisição de sinal de acelerometria, conforme a figura 27, foi obtido o endereço MAC do BITalino, a taxa de amostragem de 1 kHz, e realizado uma leitura inicial de 100 amostras.

Figura 27: Python leitura sensor ACC.

```
import bitalino
import numpy
import matplotlib.pyplot as plt

macAddress = "20:18:06:13:03:69"

device = bitalino.BITalino(macAddress)
SamplingRate=1000    # taxa de amostragem de 1000Hz
device.battery(0)
device.start(SamplingRate, [4])# Sensor 4(A5)

data = device.read(nSamples=100) # Leitura de 100 amostras
print(data)

plt.plot(data[:,5], 'blue', label = 'Sinal A5')
plt.title("Sinal ACC")
plt.xlabel("Amostras")
plt.ylabel("Nº em Bits")
plt.show()

device.stop()
device.close()
```

Fonte: Elaborado pelo Autor.

Ao iniciar a aquisição, obtivemos a leitura do sensor ACC com os seguintes valores, conforme ilustrado na figura 28. Como o sensor estava inicialmente em repouso, os valores apresentam pouca variação, permanecendo na faixa de valores entre 610 até 630 aproximadamente.

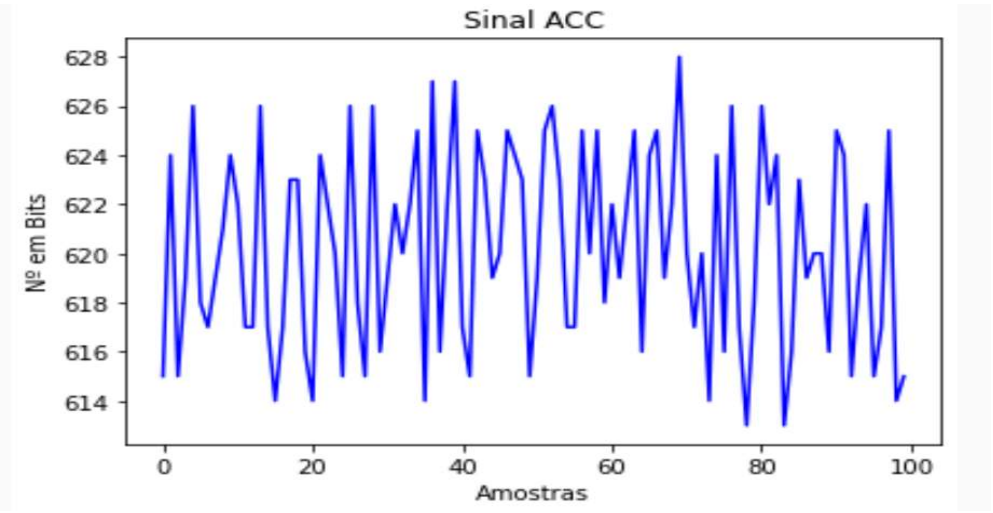
Figura 28: Aquisição sinal ACC Python

```
In [2]: runfile('C:/Users/luc
[[  0   1   1   0   0 615]
 [  1   1   1   0   0 624]
 [  2   1   1   0   0 615]
 [  3   1   1   0   0 619]
 [  4   1   1   0   0 626]
 [  5   1   1   0   0 618]
 [  6   1   1   0   0 617]
 [  7   1   1   0   0 619]
 [  8   1   1   0   0 621]
 [  9   1   1   0   0 624]
 [ 10   1   1   0   0 622]
 [ 11   1   1   0   0 617]
 [ 12   1   1   0   0 617]
 [ 13   1   1   0   0 626]
 [ 14   1   1   0   0 617]
 [ 15   1   1   0   0 614]
 [  0   1   1   0   0 617]
 [  1   1   1   0   0 623]
 [  2   1   1   0   0 623]
 [  3   1   1   0   0 616]
```

Fonte: Elaborado pelo Autor.

Após a aquisição das 100 amostras, plotamos o gráfico na figura 29 para observamos se os valores obtidos condizem com o sensor na posição de repouso. Foi verificado que os valores estão próximos aos obtidos pelo software OpenSignals.

Figura 29: Exemplo gráfico sinal ACC com 100 amostras



Fonte: Elaborado pelo Autor.

4.5 Segunda versão supervisório para PC

Na segunda versão do supervisório, desenvolvemos a interface gráfica do sistema. O mesmo contará com botões, gráficos, opções de seleção de gráficos, entre outros elementos inserido para a parte visual do sistema.

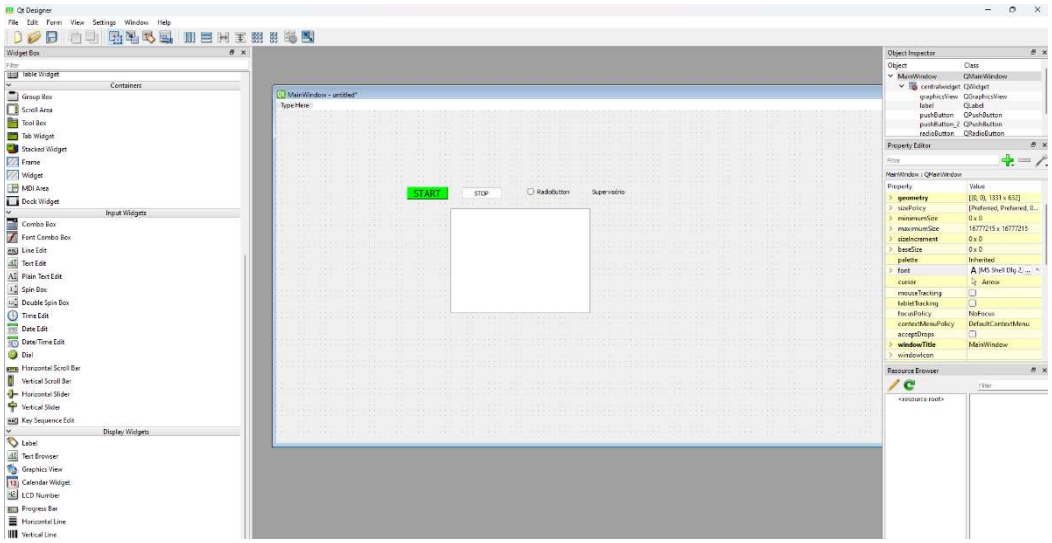
4.5.1 Qt Designer

O Qt Designer é a ferramenta Qt para projetar e construir interfaces gráficas de usuário (GUIs) com Qt *Widgets*. Com a Qt Designer, podemos compor e personalizar janelas, caixas de diálogo, entre outros elementos de interface de uma maneira que possamos testá-los usando diferentes estilos e resoluções. Os *Widgets* criados com o Qt Designer foram integrados com o código programado em Python. Além disso, eles também podem ser alterados dinamicamente por meio do código, também é possível utilizar ícones e plug-ins personalizados para aprimorar a visualização do sistema. A seguir, vamos verificar a montagem da *dashboard* do sistema supervisório.

4.5.2 Interface gráfica do supervisório

Com o Qt Designer temos várias opções de interfaces gráficas, como adicionar botão, adicionar uma seleção, um *label*, gráficos entre outras *Widgets* disponíveis no software. Na imagem abaixo podemos observar alguns dos widgets utilizado para a criação da interface gráfica.

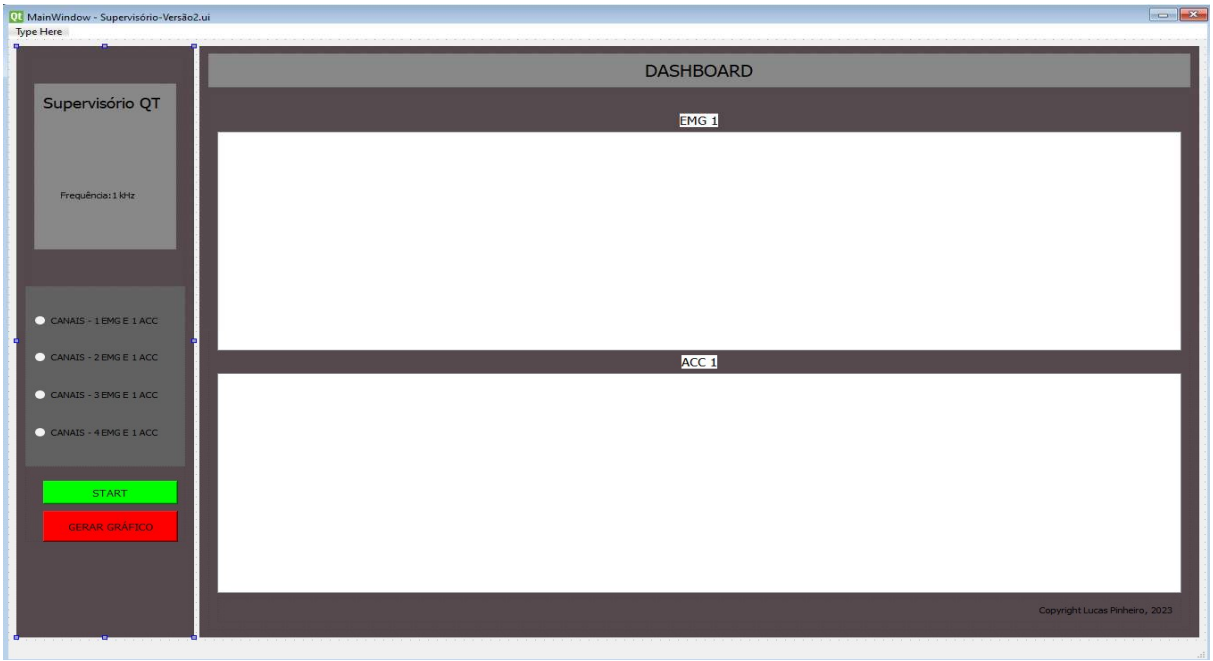
Figura 30: Interface do Qt Designer



Fonte: Elaborado pelo Autor.

A interface gráfica do sistema possui dois botões, start e gerar gráfico, start para iniciar a aquisição dos sinais e gerar gráfico para mostrar os sinais nos gráficos, também possui as 4 opções de seleção de gráficos, conforme a figura 31 da interface inicial do supervisório.

Figura 31: Interface Gráfica Inicial do supervisório



Fonte: Elaborado pelo Autor.

4.5.3 Chamando a interface gerada em Python

Quando criamos a interface no software Qt Designer temos um arquivo .ui, para isso colocamos na mesma pasta do código e carregamos a interface. Na figura 32 podemos ver como foi carregada a interface para o programa Spyder, também é possível transformar o arquivo ui em Python e trabalhar diretamente dele, mas para ter um código de mais fácil visualização vamos apenas carregar ele no nosso código.

Figura 32: Python carregando a interface

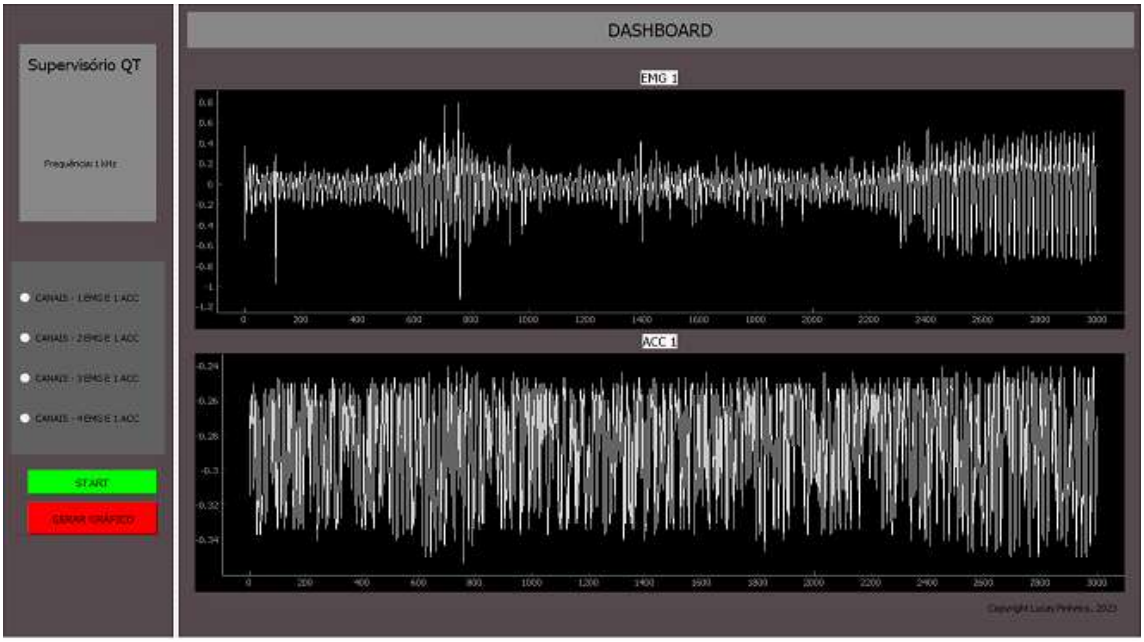
```
from PyQt5 import uic, QtCore, QtGui, QtWidgets
from pyqtgraph import PlotWidget
import bitalino
import numpy as np
import time

macAddress = "20:18:06:13:03:69"

device = bitalino.BITalino(macAddress)
SamplingRate=1000     # taxa de amostragem de 1000Hz
device.battery(0)

app=QtWidgets.QApplication([])
supervisorio=uic.loadUi("Supervisório-Versão2.ui")
supervisorio.show()
app.exec()    # Executa aplicação
```

Fonte: Elaborado pelo Autor.

4.5.4 Aquisição sinal na Interface

Com o arquivo carregado vamos utilizar a interface no código e chamar cada uma das opções. No apêndice C1 pode ser visto como ficou o código inicial, que foi transformado os valores de 10 bits (0 a 1023) para valores de tensão de -1,65 mV até 1,65 mV. Iniciamos a aquisição dos sinais e obtivemos na figura 33 os sinais em um intervalo de 3000 amostras, apenas a fim de exemplo.

Figura 33: Interface supervisório com gráficos



Fonte: Elaborado pelo Autor.

Podemos observar que os sinais ainda não estão com o filtro passa-baixa nem com processos de pré-processamento para rotulagem utilizado nos resultados, que são usados para remover os ruídos indesejáveis do sinal e obter uma melhor visualização nos gráficos.

4.6 Terceira versão do supervisório

Na terceira versão, utilizamos todos os canais, sendo 4 canais para eletromiografia (EMG), 1 canal para acelerometria (ACC) e 1 canal para a referência dos eletrodos dos sensores EMG. Esses canais foram configurados no kit BITalino *MuscleBIT*, que é um kit pré-montado projetado para medir a atividade muscular por meio da aquisição de sinais de eletromiografia. Conforme ilustrado na figura 34, os 4 sensores de EMG foram conectados nos canais A1 até A4 do Bitalino, enquanto a referência dos eletrodos foi conectada ao canal A5. O sensor ACC foi colocado no canal A6 do BITalino.

Figura 34: Posição eletrodos EMG

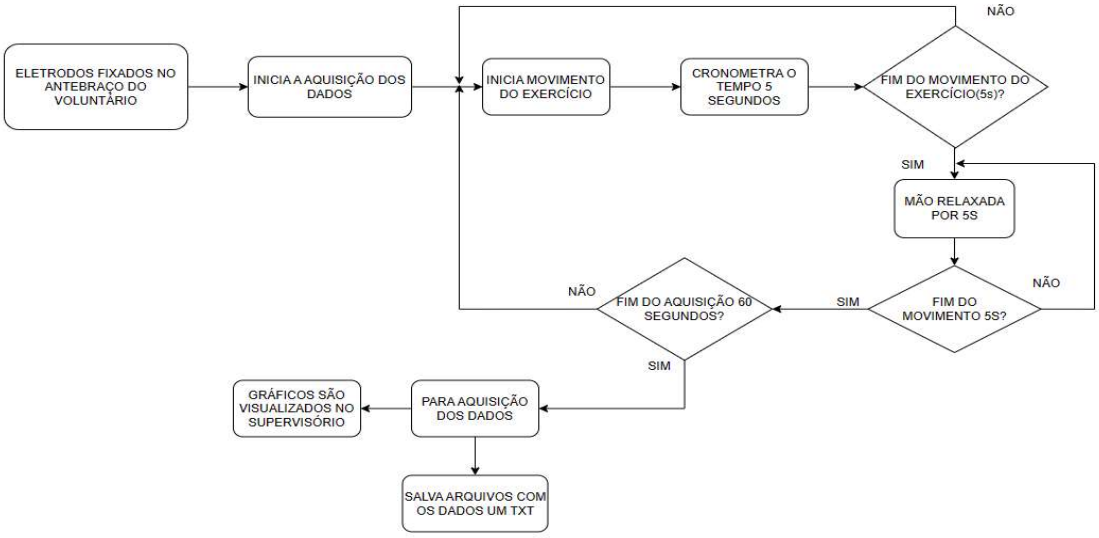
Fonte: Elaborado pelo Autor.

Para realizar a aquisição dos sinais de EMG, é necessário seguir os seguintes procedimentos conforme a SENIAM (*Surface ElectroMyoGraphy for the Non-Invasive Assessment of Muscles*) para a fixação dos eletrodos na região de interesse:

a) O voluntário deve sentar-se em uma cadeira e apoiar os dois braços sobre a mesa;
b) O voluntário limpa a região da pele a ser colocado os eletrodos com álcool isopropílico 70%, de preferência depilada;
c) Após inicia a etapa de fixação dos eletrodos de superfícies ao redor do antebraço direito ou esquerdo, utilizando 4 pares de eletrodos de superfície Ag/AgCl;
d) O eletrodo de referência deve ser conectado na região do cotovelo e, em seguida, conectado ao BITalino no canal A5;
e) Conectar os quatros eletrodos restantes nos canais A1 a A4;
f) Por fim, o voluntário inicia os movimentos para aquisição do sinal EMG;

Após a fixação dos eletrodos, é iniciado a aquisição dos sinais EMG, conforme mostrado no fluxograma da imagem 35.

Figura 35: Fluxograma aquisição sinal EMG



Fonte: Elaborado pelo Autor.

Para a aquisição dos sinais EMG, cada movimento foi repetido seis vezes, com um tempo de descanso de 5 segundos entre eles, totalizando 60 segundos, como baseado por (ATZORI et al., 2014a e ATZORI et al., 2015). Foi analisado a execução de três exercícios diferentes: mão estendida, mão fechada e polegar para cima, conforme podemos observar na figura 36.

Figura 36: Exercícios para aquisição sinal EMG

1	Mão estendida	
2	Mão fechada	
3	Polegar para cima	
4	Mão relaxada	

Fonte: Elaborado pelo Autor.

Na versão final do supervisório para PC, foi incorporado os 4 canais EMG e 1 canal ACC. O software utilizado para a interface gráfico foi o Qt Designer, além dos botões e textos visto na segunda versão do supervisório, foi adicionado um timer para monitorar o tempo em segundos dos exercícios analisados, e colocado os 5 gráficos que serão utilizados para análises. No supervisório, cada gráfico é representado por uma cor, sendo que o sinal EMG A1 é representado na cor azul, o sinal EMG A2 pela cor vermelha, o sinal EMG A3 pela cor verde, o sinal EMG A4 pela cor laranja, o sinal ACC A6 pela cor roxa. Cada gráfico possui uma escala, com o eixo x representando o número de amostras e o eixo y representando os valores de tensão, que variam de -1,65 mV até 1,65 mV.

Na figura 37 é possível observar a última versão do supervisório, tendo as 4 opções de seleção de canal que serão exibidas, o botão start que inicia a aquisição dos sinais, durante o intervalo de tempo de 60 segundos, ao mesmo tempo, o supervisório escreve e armazena em um arquivo de texto os valores obtidos, com isso permitindo a análise dos dados obtidos e a identificação dos movimentos realizados. Ao clicarmos no gerar gráfico, é mostrado todos os 5 gráficos obtidos durante os 60 segundos. O código do Qt Designer transformado em Python pode ser visto no apêndice A.

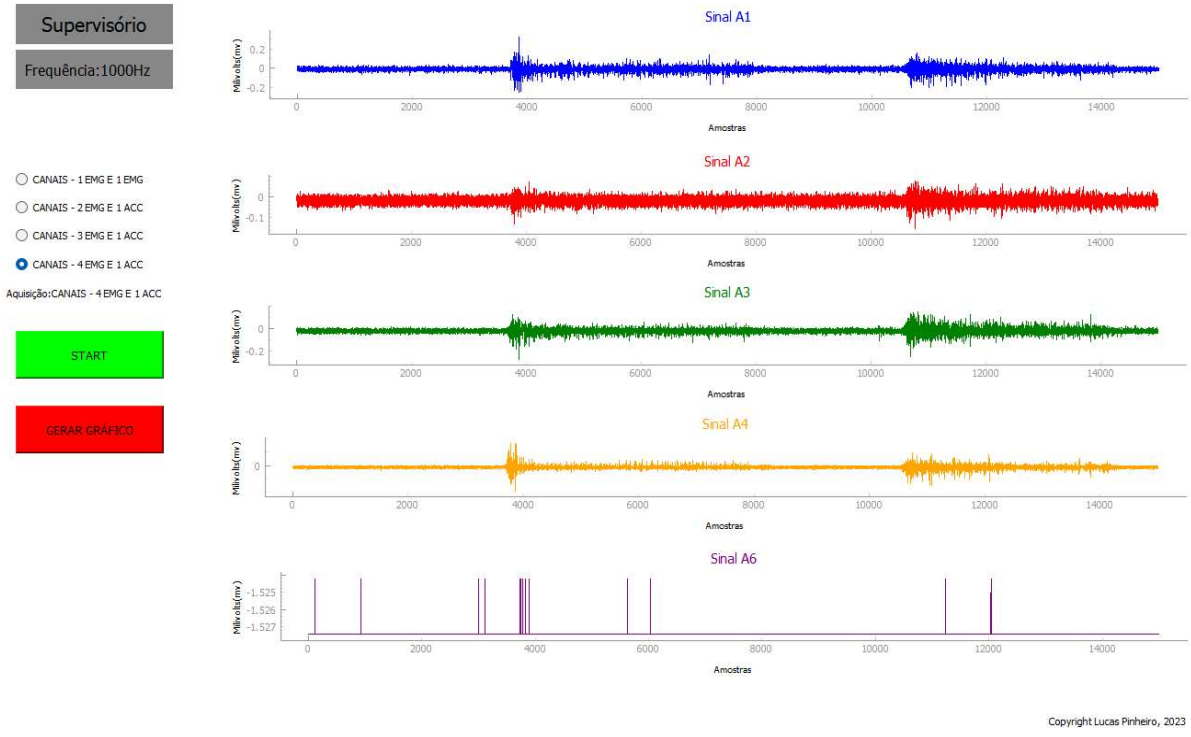
Figura 37: Supervisório terceira versão

Fonte: Elaborado pelo Autor.

5 RESULTADOS

No trabalho foi desenvolvido um supervisório para PC com o objetivo de coletar e armazenar sinais de eletromiografia (EMG) e sinais de acelerometria (ACC) do sistema embarcado BITalino, e apresentar no supervisório para PC funções gráficas para visualização dos sinais. A taxa de atualização do sinal foi fixada em 1 kHz. Para a análise dos resultados, foi realizada a execução de três exercícios diferentes: mão estendida, mão fechada e polegar para cima, com o objetivo de analisar o comportamento dos sinais. Na figura 38, é possível visualizar o resultado do supervisório, que exibe a interface com os 4 gráficos EMG, o gráfico de ACC, os botões e as opções de seleção de canal.

Figura 38: Supervisório para PC

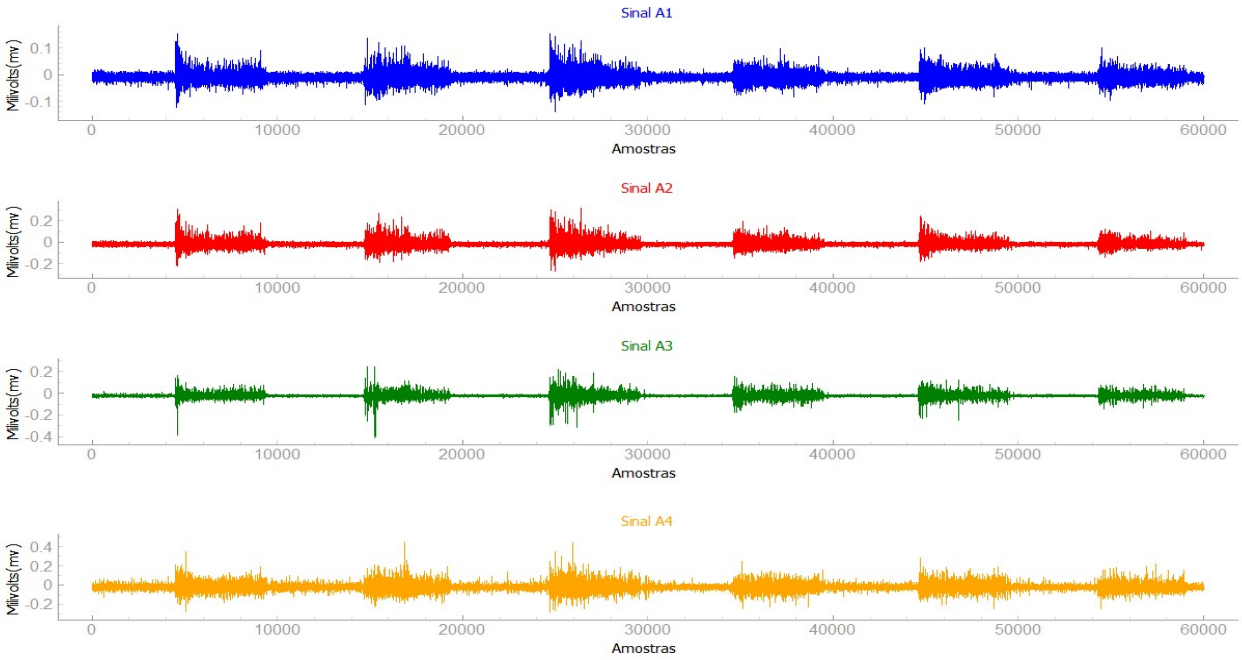


Fonte: Elaborado pelo Autor.

5.1 Aquisição sinal EMG exercício 1

Para a aquisição do sinal do exercício 1, consideramos um tempo de aproximadamente 60 segundos. Iniciamos com a mão relaxada durante 5 segundos (correspondendo a cerca de 5000 amostras em média) e, em seguida, estendemos a mão por 5 segundos. Esse processo foi repetido 6 vezes, resultando em um total de aproximadamente 60.000 amostras. Na figura 39, podemos observar os sinais A1, A2, A3 e A4, sendo possível notar que ao realizar esse exercício da mão fechada, os sinais apresentam um aumento nos valores registrados. Os valores do sinal variam de 0 a 1023 bits, o que corresponde a uma faixa de -1,65 mV a 1,65 mV. Durante o exercício, os valores oscilam entre 510 e 530, indicando uma resposta do sistema aos estímulos motores. É importante observar que alguns sinais apresentam uma amplitude maior quando a mão está relaxada, enquanto outros apresentam uma amplitude menor, especificamente, os sinais A2 e A3 exibem uma visualização mais nítida dos padrões de sinal, pois mantém o sinal mais próximo a 0 volts quando a mão está relaxada.

Figura 39: Aquisição Sinal EMG exercício 1

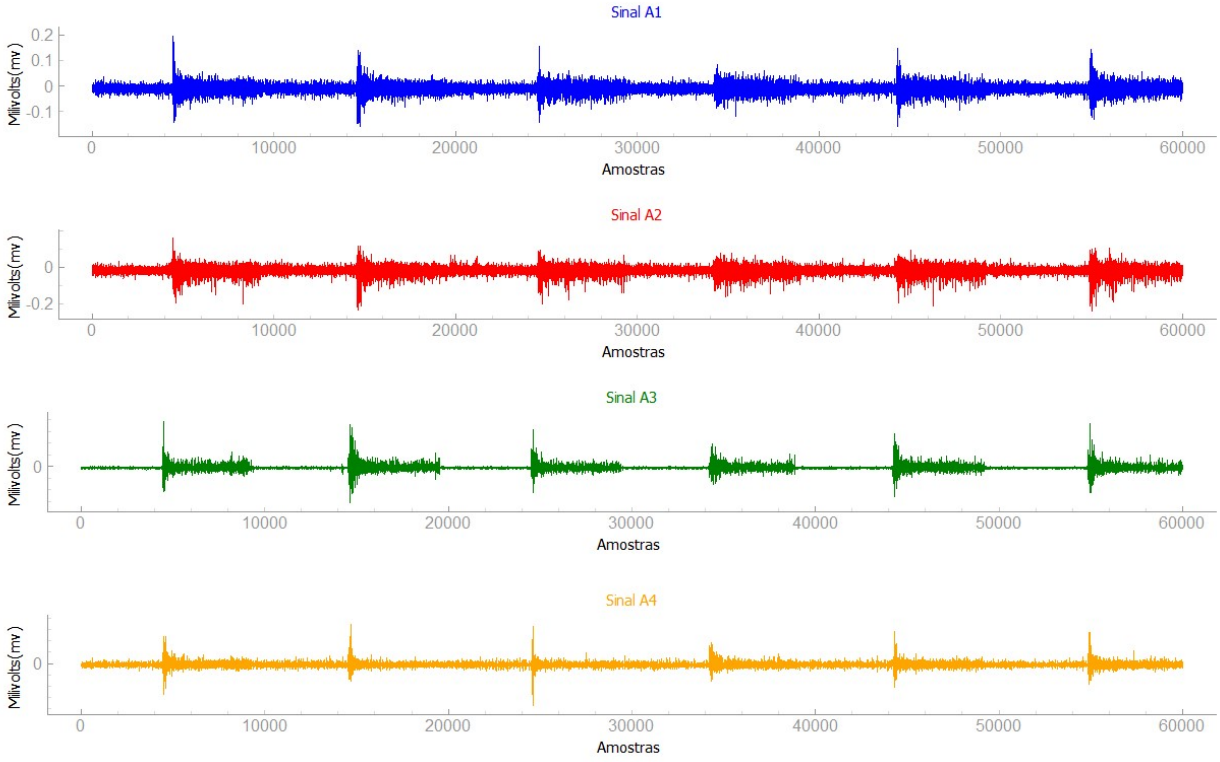


Fonte: Elaborado pelo Autor.

5.2 Aquisição sinal EMG exercício 2

No exercício 2, a aquisição do sinal seguiu os mesmos procedimentos do exercício 1. O tempo total de aquisição foi de 60 segundos, sendo os 5 primeiros segundos com a mão relaxada, seguidos por 5 segundos mantendo a mão fechada. Esse ciclo foi repetido 6 vezes, resultando em um total aproximado de 60.000 amostras de sinal. A figura 40 apresenta a variação do sinal ao longo da execução do exercício 2. Analisando os gráficos, observou-se que o sinal capturado pelo canal A3 demonstrou uma onda mais próxima do padrão esperado, facilitando a identificação do movimento realizado.

Figura 40: Aquisição sinal EMG exercício 2



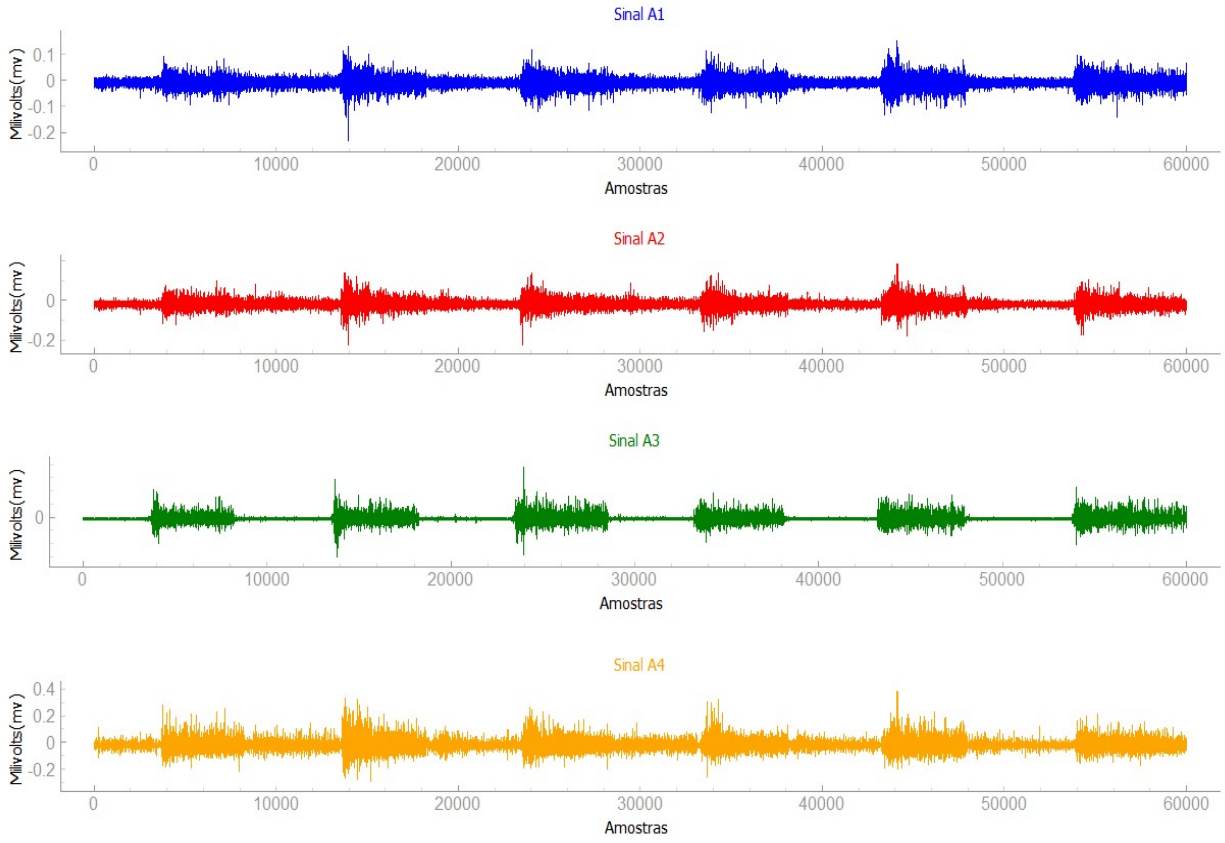
Fonte: Elaborado pelo Autor.

5.3 Aquisição sinal EMG exercício 3

No exercício 3, seguimos os mesmos procedimentos dos exercícios 1 e 2 para a aquisição do sinal. O tempo total de aquisição foi de 60 segundos, sendo os primeiros 5 segundos com a mão relaxada e, em seguida, mantivemos o polegar para cima por mais 5 segundos. Esse ciclo foi repetido 6 vezes, totalizando

aproximadamente 60.000 amostras de sinal. A figura 41 representa a variação do sinal durante a execução do exercício. Ao analisar os dados, observamos a resposta do sinal de acordo com os diferentes movimentos realizados e novamente o sinal EMG A3 apresentou uma melhor forma de onda, possuindo uma facilidade maior para a identificação do gesto.

Figura 41: Aquisição sinal EMG exercício 3



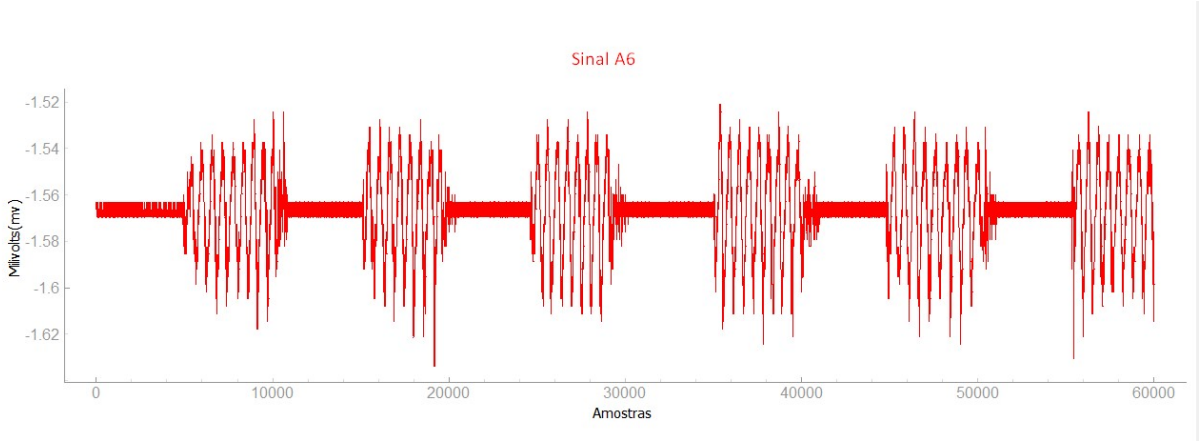
Fonte: Elaborado pelo Autor.

5.4 Exercício aquisição sensor ACC

Para realizar a análise da aquisição do sensor ACC, foi realizado o exercício de movimentar para cima e para baixo, com o objetivo de criar um movimento e observar a variação do sinal registrado. A figura 42 ilustra essa variação, na qual podemos observar a formação de uma senoide ao movimentarmos o sensor. A faixa de variação do sinal de tensão ACC varia de -1,65 mV até 1,52 mV. É importante destacar que, diferentemente da aquisição dos sinais EMG, a resolução do conversor

analógico-digital (A/D) utilizado para o ACC é de 6 bits, o que resulta em valores no intervalo de 0 a 63. Essas informações são relevantes para compreender a capacidade de detecção e a precisão do sinal capturado pelo sensor ACC.

Figura 42: Exercício aquisição sinal ACC



Fonte: Elaborado pelo Autor.

5.5 Pré-processamento dos sinais

O pré-processamento prévio dos sinais EMG envolveu três etapas: filtragem, retificação e suavização de cada sinal. Na etapa da filtragem para eliminar ruídos de alta frequência, os biosinais foram submetidos a um filtro passa-baixa *Butterworth* de segunda ordem, com uma frequência de corte de 300 Hz. Após essa etapa, os sinais foram retificados e utilizado um filtro de média móvel para a suavização dos sinais.

A retificação foi uma etapa crucial para obter a forma ou envelope do sinal EMG, pois o sinal EMG, em essência, possui um valor médio próximo de zero (ROSE, 2011). A retificação escolhida neste trabalho foi a de onda completa para manter toda a energia do sinal. A função valor absoluto (equação 1) foi utilizada para retificar os sinais de cada um dos canais EMG, transformando os valores negativos em positivos.

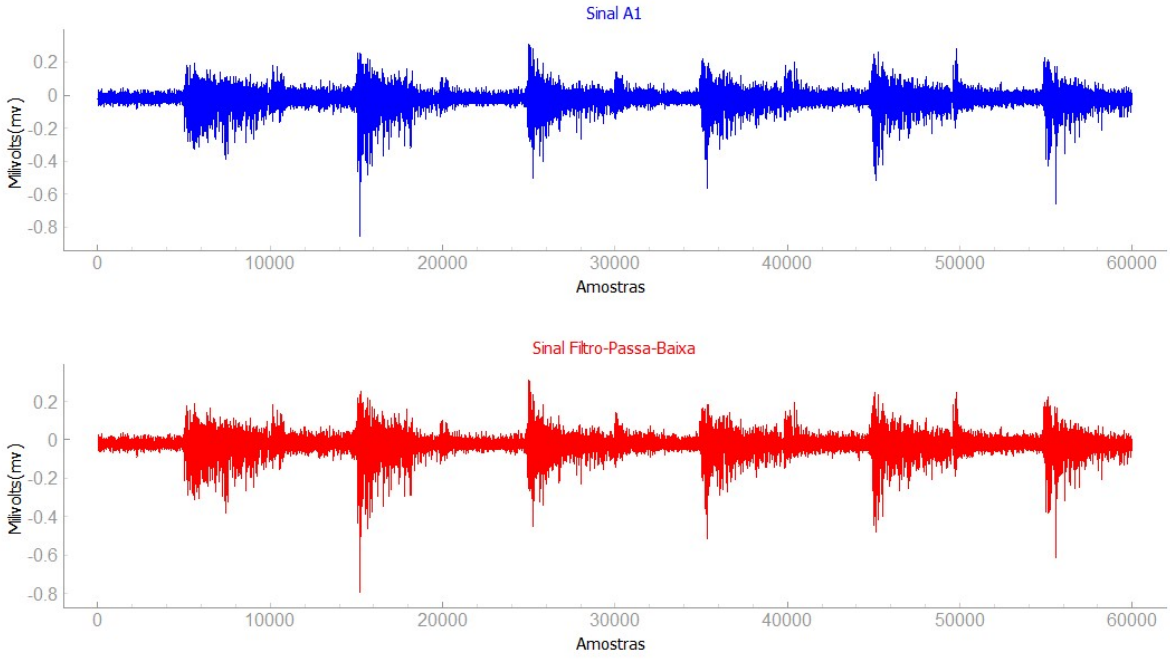
$$EMG_{RET} = |EMG| \tag{1}$$

Na etapa de aplicar uma suavização no sinal retificado, utilizou-se um filtro passa-baixa como um filtro de média móvel, que calcula a média dos valores do sinal em uma janela e substitui cada valor pelo valor médio correspondente, ajudando a reduzir ruído e melhorar o sinal para fácil identificação dos movimentos.

5.5.1 Filtro passa-baixa

O filtro passa-baixa serviu para reduzir frequências mais altas que contêm ruídos e são indesejadas, sendo utilizado um filtro passa-baixa *Butterworth* de segunda ordem, com uma frequência de corte de 300 Hz. Para a aplicação do filtro na onda foi utilizado a filtragem filtfilt, que é uma função da biblioteca *scipy,* a qual possibilita a aplicação de um filtro digital. A função filtfilt é uma ferramenta eficaz para filtragem, pois aplica um filtro digital linear duas vezes, uma vez para frente e outra para trás. Na figura 43, é possível observar o resultado da aplicação do filtro passa-baixa no sinal selecionado. Em comparação ao sinal inicial, foi difícil perceber diferença, pois a resolução exibida no gráfico é pequena, porém há uma atenuação nas frequências indesejadas, tais como ruídos. O código do filtro pode ser visto no apêndice B1.

Figura 43: Filtro passa-baixa sinal EMG



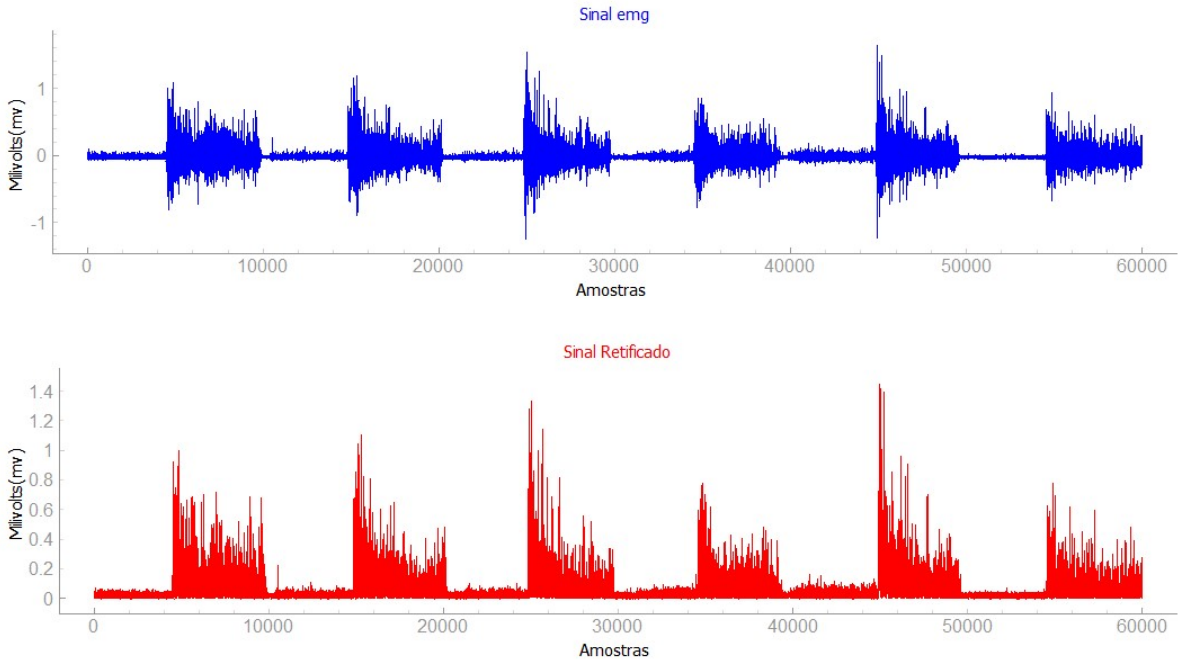
Fonte: Elaborado pelo Autor.

5.5.2 Retificação

O processo de retificação consiste em transformar o sinal em valores acima do zero, para isso foi utilizado a função np.abs conforme apêndice B2, ao qual converte todos os valores negativos do sinal em valores positivos. Ao aplicar a função np.abs

ao sinal, obtém-se uma forma de onda retificada na qual todos os valores negativos são convertidos em seus respectivos valores positivos. Essa etapa é utilizada em diferentes aplicações, para obter uma representação apenas positiva do sinal e analisar apenas a magnitude do sinal. Na figura 44, é possível visualizar o resultado do processo de retificação aplicado ao sinal EMG, evidenciando a transformação dos valores negativos em valores positivos ao longo da forma de onda.

Figura 44: Sinal EMG A1 retificado

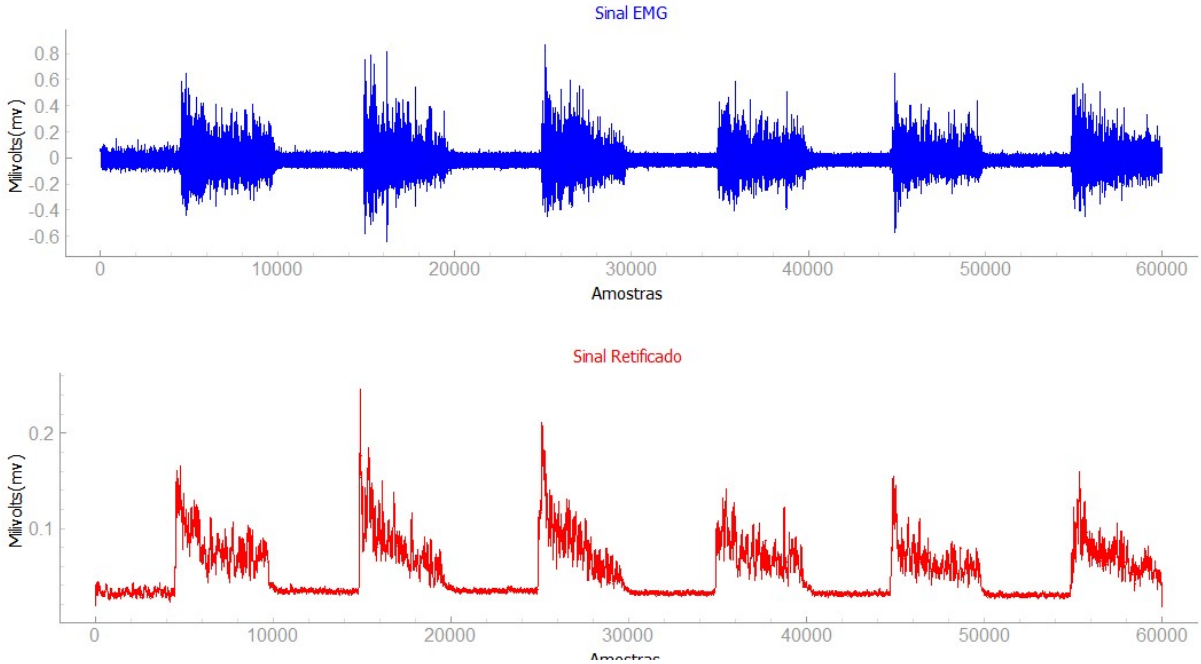


Fonte: Elaborado pelo Autor.

5.5.3 Suavização

Com o objetivo de melhorar o sinal e facilitar a identificação do sinal, foi aplicada uma técnica de suavização. Essa etapa tem como finalidade reduzir o ruído e as flutuações indesejadas no sinal retificado, como observamos na figura 45, proporcionando assim uma forma de onda mais legível e de fácil identificação do gesto. A suavização é aplicada por meio da aplicação de um filtro que promove a média das variações dos sinais. Na imagem 45 é possível visualizar o resultado desse processo, no qual é evidente a redução das flutuações indesejadas e a obtenção de uma forma de onda mais clara e de fácil visualização do movimento. O código do trecho pode ser visto no apêndice B3.

Figura 45: Sinal suavizado



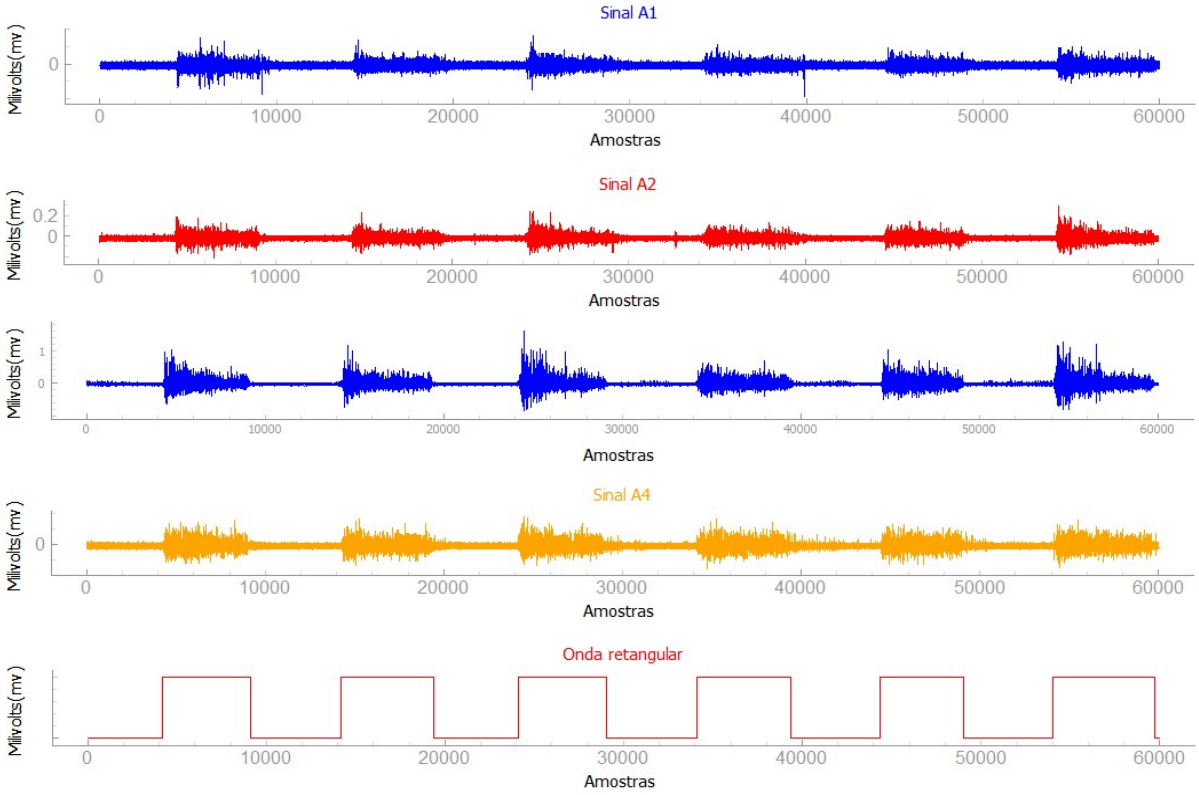
Fonte: Elaborado pelo Autor.

5.5.4 Marcação do início e fim do gesto de mão.

A fim de marcar o início e o fim do gesto de mão realizado, utilizamos uma forma de onda retangular para indicar o momento em que o movimento começou. Para essa etapa é utilizado um comparador lógico, que gera um sinal binário como base no sinal de envelope suavizado. É calculado uma média dos valores do sinal suavizado, por meio de um laço for, no qual é verificado se o valor atual do sinal é maior ou igual a 70% da média do sinal suavizado, caso seja verdadeiro, o valor 1 é adicionado a uma lista, indicando que houve um movimento e o sinal aumentou a amplitude, enquanto o valor 0 indica que o sinal está abaixo de 70% da média e não foi detectado movimento da mão.

Para essa comparação, foi utilizado o sinal que apresentava a “melhor” forma de onda para comparação, o canal A3 apresentou essa melhor forma, que em repouso o valor ficava próximo de 0 volts, com o sinal retificado e suavizado foi possível identificar claramente o início e o fim do gesto. O trecho do código do comparador pode ser visto no apêndice B4.

Figura 46: Sinal com identificação dos gestos/movimentos



Fonte: Elaborado pelo Autor.

Neste capítulo foi abordado os resultados do supervisório para PC voltado para a coleta, armazenamento e visualização de sinais de eletromiografia (EMG) e sinais de acelerometria (ACC) obtidos do sistema embarcado BITalino. O objetivo principal foi analisar o comportamento dos sinais durante a execução de três exercícios diferentes: mão estendida, mão fechada e polegar para cima, para cada exercício, foram realizadas aquisições de sinais com duração de 60 segundos, utilizando uma taxa de amostragem fixada em 1 kHz. Os sinais de EMG foram apresentados em quatro gráficos distintos sendo A1, A2, A3 e A4, e o sinal ACC no gráfico A6. Durante as análises, observou-se que os sinais de EMG apresentaram variações nos valores registrados, sendo que os sinais A2 e A3 demonstraram uma visualização mais nítida dos padrões de sinal. Em seguida, foram descritas as etapas de pré-processamento aplicadas aos sinais de EMG. Isso envolveu a filtragem para eliminar ruídos de alta frequência, a retificação para obter a forma ou envelope do sinal e a suavização para reduzir ruídos e flutuações indesejadas. Através da aplicação dessas técnicas, foi

possível obter formas de onda mais claras e legíveis, facilitando a identificação dos movimentos realizados.

As técnicas de pré-processamento aplicadas aos sinais de EMG foram eficazes para a identificação do início e do término dos gestos realizados. Os resultados obtidos fornecem uma base sólida para análises mais aprofundadas e aplicações futuras relacionadas ao monitoramento e controle de movimentos por meio de sinais de EMG e ACC.

6 CONCLUSÕES

Como objetivo do trabalho pretendeu-se desenvolver um sistema supervisório para PC que seja capaz de coletar e armazenar sinais de eletromiografia e acelerometria com uma taxa de amostragem de 1 kHz. O sistema projetado demonstrou eficiência na coleta dos sinais EMG e ACC, permitindo a visualização simultânea de até 4 sinais EMG e 1 sinal ACC. Os dados coletados foram apresentados de forma clara, fornecendo informações sobre a quantidade de amostras e a variação da tensão ao realizar os exercícios. Além disso, os dados coletados foram armazenados de forma organizada em um banco de dados, utilizando um arquivo de texto, facilitando o acesso aos dados e permitindo consultas futuras, possibilitando a realização de análises retrospectivas e comparações entre diferentes sessões de coleta de dados.

Uma etapa importante realizada no sistema nos resultados foi o pré-processamento dos sinais, que possibilitou verificar a mudança dos movimentos, através da marcação do início e fim do gesto de mão. Essa marcação foi obtida por meio das etapas de rotulagem que contribuiu em filtragem, retificação, suavização e marcação do sinal. Portanto, o sistema desenvolvido atingiu o objetivo proposto, fornecendo uma solução eficiente e abrangente para a coleta, armazenamento e visualização dos sinais de EMG e ACC através do supervisório para PC.

6.1 Sugestões para trabalhos futuros

Algumas sugestões para trabalhos futuros:

a) Utilização de uma plataforma de prototipagem rápida (exemplo: Raspberry Pi) para armazenar dados e conseguir a aquisição dos sinais em tempo real.
b) Desenvolvimento/atualização do sistema para incorporação e sincronização de 2 ou mais plataformas Bitalinos para aumento do número de canais EMG e ACC.

REFERÊNCIAS

ATZORI, M. et al. **Electromyography data for non-invasive naturally-controlled robotic hand prostheses**. 2014. Disponível em: <https://www.nature.com/articles/sdata201453.pdf>. Acesso em: 15 mai. 2023

ATZORI, M.; MÜLLER, H. Control Capabilities of Myoelectric Robotic Prostheses by Hand Amputees: A Scientific Research and Market Overview. Frontiers in Systems Neuroscience, 2015.

ATZORI, M.; GIJSBERTS, A.; CASTELLINI, C.; MÜLLER, H.; CAPUTO, B.; Classification of hand movements in amputated subjects by sEMG and accelerometers. International Conference IEEE Engineering in medicine and Biology Society, 2014b

ANACONDA DISTRIBUTION. **About Anaconda**. 2023. Disponível em: <https://www.anaconda.com/about-us>. Acesso em: 3 mar. 2023.

BALBINOT, Alexandre; BRUSAMARELLO, Valner João. **Instrumentação e Fundamentos de Medidas**. 3. ed. Rio de Janeiro: LTC, 2019. Volume 2.

BITALINO. BITalino (r)evolution Plugged Kit Data Sheet. 2016. Disponível em <https://www.bitalino.com/storage/uploads/media/revolution-bitalino-plugged-kit-datasheet.pdf>. Acesso em: 8 setem. 2022.

BITALINO. BITalino (r)evolution MuscleBIT Bundle Data Sheet. 2020. Disponível em <https://www.creact.co.jp/wp-content/uploads/2022/06/MuscleBIT-Bundle-Datasheet.pdf>. Acesso em: 8 setem. 2022.

BITALINO. Acclerometer(ACC) Sensor Data Sheet. 2020. Disponível em < <https://support.pluxbiosignals.com/wp-content/uploads/2021/11/revolution-acc-sensor-datasheet-revb.pdf>. Acesso em: 8 setem. 2022.

BITALINO. Electromyography(EMG) Assembled Sensor Datasheet. 2020. Disponível em: <https://support.pluxbiosignals.com/wp-content/uploads/2021/11/emg-sensor-datasheet-assembled-1.pdf>. Acesso em: 8 set. 2022.

CUNHA, Alessandro – "Sistemas Embarcados", Revista *Saber Eletrônica*, 414, Editora: Saber, BRASIL, 2007.

CORRÊA, Daniel dos Santos. **Desenvolvimento de um sistema para estudo da marcha humana por videogrametria e acelerometria**. Dissertação (Mestrado em Engenharia Elétrica) - Universidade Federal do Rio Grande do Sul, Porto Alegre, 2010.

COSTA, Robson. RT-WiFi: Uma Arquitetura para Comunicação de Tempo-Real em Redes IEEE 802.11 Infraestruturadas, 2013.

DENARDIN, L., BARRIQUELO, R. Sistemas operacionais de tempo real e sua aplicação em sistemas embarcados, 2019.

ESPRESSIF. ESP32. 2023. Disponível em < https://www.espressif.com/en/products/socs/esp32>. Acesso em 8 setem. 2022.

FREIXO, R. A. E. Electromyography and inertial sensor-based gesture detection and control. p. 106, 2015.

RICCIOTTI, Antonio Carlos Duarte. Utilização de wavelets no processamento de sinais EMG. 2006. 133 f. Dissertação (Mestrado em Engenharias) - Universidade Federal de Uberlândia, Uberlândia, 2006.

SENIAM. What is Seniam?. Disponível em: <http://www.seniam.org/>. Acesso em: 10 mai. 2022.

SIQUEIRA, Thiago Senador de. Bluetooth – Características, protocolos e funcionamento. Universidade Estadual de Campinas–Unicamp, 2006. Disponível em:

<http://www.ic.unicamp.br/~ducatte/mo401/1s2006/T2/057642-T.pdf.>. Acesso em 23 setem. 2022.

QT DESIGNER. Qt Designer Manual. 2023. Disponível em <https://doc.qt.io/qt-6/qtdesigner-manual.html> Acesso em 9 setem. 2022.

ROSE, William. Electromyogram Analysis. In Mathematics and Signal Processing for Biomechanics, 2019. Disponível em: https://www1.udel.edu/biology/rosewc/kaap686/notes/EMG%20analysis.pdf. Acesso em 10 maio de 2023.

TÉCNICO LISBOA. BITalino à conquista do mundo. Disponível em: https://tecnico.ulisboa.pt/pt/noticias/bitalino-a-conquista-do-mundo/#:~:text=Consiste%20num%20kit%20eletr%C3%B3nico%20que,atividade%20eletrodermal%20ou%20o%20desporto>. 2017. Acesso em: 8 setem. 2022.

APÊNDICE A – TRECHOS DE CÓDIGO REALIZADO NO QT DESIGNER

A.1 – Código Qt Designer Versão3 transformado em Python

```
from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(1538, 847)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.frame = QtWidgets.QFrame(self.centralwidget)
        self.frame.setGeometry(QtCore.QRect(9, -1, 211, 801))
        font = QtGui.QFont()
        font.setPointSize(16)
        self.frame.setFont(font)
        self.frame.setStyleSheet("background-color: rgb(85, 73, 78);\n"
"background-color: rgb(255, 255, 255);")
        self.frame.setFrameShape(QtWidgets.QFrame.StyledPanel)
        self.frame.setFrameShadow(QtWidgets.QFrame.Raised)
        self.frame.setObjectName("frame")
        self.frame_3 = QtWidgets.QFrame(self.frame)
        self.frame_3.setGeometry(QtCore.QRect(10, 10, 191, 111))
        font = QtGui.QFont()
        font.setPointSize(16)
        self.frame_3.setFont(font)
        self.frame_3.setStyleSheet("")
        self.frame_3.setFrameShape(QtWidgets.QFrame.StyledPanel)
        self.frame_3.setFrameShadow(QtWidgets.QFrame.Raised)
        self.frame_3.setObjectName("frame_3")
        self.verticalLayout_3 = QtWidgets.QVBoxLayout(self.frame_3)
        self.verticalLayout_3.setObjectName("verticalLayout_3")
        self.label = QtWidgets.QLabel(self.frame_3)
        font = QtGui.QFont()
        font.setPointSize(16)
        self.label.setFont(font)
        self.label.setStyleSheet("background-color: rgb(135, 135, 135);")
        self.label.setObjectName("label")
        self.verticalLayout_3.addWidget(self.label)
        self.frame_5 = QtWidgets.QFrame(self.frame_3)
        self.frame_5.setStyleSheet("background-color: rgb(85, 0, 255);\n"
"background-color: rgb(255, 255, 255);\n"
"background-color: rgb(136, 136, 136);")
        self.frame_5.setFrameShape(QtWidgets.QFrame.StyledPanel)
        self.frame_5.setFrameShadow(QtWidgets.QFrame.Raised)
        self.frame_5.setObjectName("frame_5")
        self.verticalLayout_4 = QtWidgets.QVBoxLayout(self.frame_5)
        self.verticalLayout_4.setObjectName("verticalLayout_4")
        self.label_2 = QtWidgets.QLabel(self.frame_5)
        font = QtGui.QFont()
        font.setPointSize(12)
        self.label_2.setFont(font)
        self.label_2.setStyleSheet("background-color: rgb(136, 136, 136);")
        self.label_2.setObjectName("label_2")
        self.verticalLayout_4.addWidget(self.label_2)
        self.verticalLayout_3.addWidget(self.frame_5)
        self.frame_4 = QtWidgets.QFrame(self.frame)
        self.frame_4.setGeometry(QtCore.QRect(10, 170, 181, 151))
        font = QtGui.QFont()
        font.setPointSize(16)
        self.frame_4.setFont(font)
        self.frame_4.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.frame_4.setFrameShape(QtWidgets.QFrame.StyledPanel)
        self.frame_4.setFrameShadow(QtWidgets.QFrame.Raised)
        self.frame_4.setObjectName("frame_4")
        self.radioButton_1 = QtWidgets.QRadioButton(self.frame_4)
```

Fonte: Elaborado pelo Autor.

A2 – Continuação código Qt Designer transformado em Python

```
        self.radioButton_1.setGeometry(QtCore.QRect(10, 30, 151, 17))
        self.radioButton_1.setStyleSheet("")
        self.radioButton_1.setObjectName("radioButton_1")
        self.radioButton_2 = QtWidgets.QRadioButton(self.frame_4)
        self.radioButton_2.setGeometry(QtCore.QRect(10, 60, 151, 17))
        self.radioButton_2.setObjectName("radioButton_2")
        self.radioButton_3 = QtWidgets.QRadioButton(self.frame_4)
        self.radioButton_3.setGeometry(QtCore.QRect(10, 90, 159, 17))
        self.radioButton_3.setObjectName("radioButton_3")
        self.radioButton_4 = QtWidgets.QRadioButton(self.frame_4)
        self.radioButton_4.setGeometry(QtCore.QRect(10, 120, 159, 17))
        self.radioButton_4.setObjectName("radioButton_4")
        self.pushButton = QtWidgets.QPushButton(self.frame)
        self.pushButton.setGeometry(QtCore.QRect(20, 370, 161, 51))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.pushButton.setFont(font)
        self.pushButton.setStyleSheet("background-color: rgb(0, 255, 0);")
        self.pushButton.setObjectName("pushButton")
        self.pushButton_2 = QtWidgets.QPushButton(self.frame)
        self.pushButton_2.setGeometry(QtCore.QRect(20, 450, 161, 51))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.pushButton_2.setFont(font)
        self.pushButton_2.setStyleSheet("background-color: rgb(255, 0, 0);")
        self.pushButton_2.setObjectName("pushButton_2")
        self.label_6 = QtWidgets.QLabel(self.frame)
        self.label_6.setGeometry(QtCore.QRect(10, 319, 181, 21))
        self.label_6.setText("")
        self.label_6.setObjectName("label_6")
        self.label_4 = QtWidgets.QLabel(self.frame)
        self.label_4.setGeometry(QtCore.QRect(10, 520, 61, 31))
        font = QtGui.QFont()
        font.setPointSize(14)
        self.label_4.setFont(font)
        self.label_4.setObjectName("label_4")
        self.frame_2 = QtWidgets.QFrame(self.centralwidget)
        self.frame_2.setGeometry(QtCore.QRect(213, -4, 1121, 801))
        font = QtGui.QFont()
        font.setPointSize(16)
        self.frame_2.setFont(font)
        self.frame_2.setStyleSheet("background-color: rgb(85, 73, 78);\n"
"background-color: rgb(255, 255, 255);")
        self.frame_2.setFrameShape(QtWidgets.QFrame.StyledPanel)
        self.frame_2.setFrameShadow(QtWidgets.QFrame.Raised)
        self.frame_2.setObjectName("frame_2")
        self.frame_7 = QtWidgets.QFrame(self.frame_2)
        self.frame_7.setGeometry(QtCore.QRect(10, 10, 1111, 761))
        self.frame_7.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.frame_7.setFrameShape(QtWidgets.QFrame.StyledPanel)
        self.frame_7.setFrameShadow(QtWidgets.QFrame.Raised)
        self.frame_7.setObjectName("frame_7")
        self.graphicsView_1 = PlotWidget(self.frame_7)
        self.graphicsView_1.setGeometry(QtCore.QRect(40, 10, 1041, 141))
        self.graphicsView_1.setStyleSheet("background-color: rgb(255, 255, 255);\n"
"")
        self.graphicsView_1.setInteractive(True)
        self.graphicsView_1.setObjectName("graphicsView_1")
        self.graphicsView_2 = PlotWidget(self.frame_7)
        self.graphicsView_2.setGeometry(QtCore.QRect(39, 165, 1041, 131))
        self.graphicsView_2.setMinimumSize(QtCore.QSize(0, 79))
        self.graphicsView_2.setStyleSheet("\n"
"background-color: rgb(255, 255, 255);")
        brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
        brush.setStyle(QtCore.Qt.NoBrush)
```

Fonte: Elaborado pelo Autor.

A3 – Continuação código Qt Designer transformado em Python

```
        self.graphicsView_2.setBackgroundBrush(brush)
        brush = QtGui.QBrush(QtGui.QColor(0, 255, 127))
        brush.setStyle(QtCore.Qt.NoBrush)
        self.graphicsView_2.setForegroundBrush(brush)
        self.graphicsView_2.setObjectName("graphicsView_2")
        self.graphicsView_3 = PlotWidget(self.frame_7)
        self.graphicsView_3.setGeometry(QtCore.QRect(39, 305, 1041, 131))
        self.graphicsView_3.setStyleSheet("\n"
"background-color: rgb(255, 255, 255);")
        brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
        brush.setStyle(QtCore.Qt.NoBrush)
        self.graphicsView_3.setBackgroundBrush(brush)
        self.graphicsView_3.setObjectName("graphicsView_3")
        self.graphicsView_4 = PlotWidget(self.frame_7)
        self.graphicsView_4.setGeometry(QtCore.QRect(39, 446, 1041, 131))
        self.graphicsView_4.setStyleSheet("background-color: rgb(255, 255, 255);")
        brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
        brush.setStyle(QtCore.Qt.NoBrush)
        self.graphicsView_4.setBackgroundBrush(brush)
        brush = QtGui.QBrush(QtGui.QColor(255, 85, 127))
        brush.setStyle(QtCore.Qt.NoBrush)
        self.graphicsView_4.setForegroundBrush(brush)
        self.graphicsView_4.setObjectName("graphicsView_4")
        self.graphicsView_5 = PlotWidget(self.frame_7)
        self.graphicsView_5.setGeometry(QtCore.QRect(40, 590, 1041, 141))
        self.graphicsView_5.setStyleSheet("background-color: rgb(255, 255, 255);")
        brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
        brush.setStyle(QtCore.Qt.NoBrush)
        self.graphicsView_5.setBackgroundBrush(brush)
        brush = QtGui.QBrush(QtGui.QColor(255, 0, 0))
        brush.setStyle(QtCore.Qt.NoBrush)
        self.graphicsView_5.setForegroundBrush(brush)
        self.graphicsView_5.setObjectName("graphicsView_5")
        self.label_3 = QtWidgets.QLabel(self.frame_2)
        self.label_3.setGeometry(QtCore.QRect(940, 780, 149, 21))
        self.label_3.setObjectName("label_3")
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 1538, 21))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "MainWindow"))
        self.label.setText(_translate("MainWindow", "    Supervisório"))
        self.label_2.setText(_translate("MainWindow", "Frequência:1000Hz"))
        self.radioButton_1.setText(_translate("MainWindow", "CANAIS - 1 EMG E 1 ACC"))
        self.radioButton_2.setText(_translate("MainWindow", "CANAIS - 2 EMG E 1 ACC"))
        self.radioButton_3.setText(_translate("MainWindow", "CANAIS - 3 EMG E 1 ACC"))
        self.radioButton_4.setText(_translate("MainWindow", "CANAIS - 4 EMG E 1 ACC"))
        self.pushButton.setText(_translate("MainWindow", "START"))
        self.pushButton_2.setText(_translate("MainWindow", "GERAR GRAFICO"))
        self.label_4.setText(_translate("MainWindow", "Timer:"))
        self.label_3.setText(_translate("MainWindow", "Copyright Lucas Pinheiro, 2023"))
from pyqtgraph import PlotWidget


```

Fonte: Elaborado pelo Autor.

A4 – Continuação código Qt Designer transformado em Python

```
if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())

```

Fonte: Elaborado pelo Autor.

APÊNDICE B – TRECHOS DE CÓDIGO REALIZADO NO PROCESSO DE ENVOLTÓRIA

B1 - Código aplicação filtro passa baixa

```
EMG_envol = Sinal_emgA1_mv
nyq = 0.5 * SamplingRate
low = 300 / nyq
b1, a1 = butter(2, low, btype='low')
EMG_envol = signal.filtfilt(b1, a1, EMG_envol)

supervisorio.graphicsView_5.setTitle("Sinal Filtro-Passa-Baixa", color="red")
styles = {'color':'r', 'font-size':'15px'}
supervisorio.graphicsView_5.setLabel('left', 'Milivolts(mv)', **styles)
supervisorio.graphicsView_5.setLabel('bottom', 'Amostras', **styles)
axis_left = supervisorio.graphicsView_5.getAxis('left')
axis_bottom = supervisorio.graphicsView_5.getAxis('bottom')
axis_left.setStyle(tickFont=QtGui.QFont("Arial", 12))
axis_bottom.setStyle(tickFont=QtGui.QFont("Arial", 12))
supervisorio.graphicsView_5.plot(EMG_envol, pen=('red'))
```

Fonte: Elaborado pelo Autor.

B2- Código retificação envoltória

```
EMG_envol_retif=np.abs(EMG_envol)
supervisorio.graphicsView_5.setTitle("Sinal Retificado", color="red")
styles = {'color':'r', 'font-size':'15px'}
supervisorio.graphicsView_5.setLabel('left', 'Milivolts(mv)', **styles)
supervisorio.graphicsView_5.setLabel('bottom', 'Amostras', **styles)
axis_left = supervisorio.graphicsView_5.getAxis('left')
axis_bottom = supervisorio.graphicsView_5.getAxis('bottom')
axis_left.setStyle(tickFont=QtGui.QFont("Arial", 12))
axis_bottom.setStyle(tickFont=QtGui.QFont("Arial", 12))
supervisorio.graphicsView_5.plot(EMG_envol_retif, pen=('red'))
```

Fonte: Elaborado pelo Autor.

B3 – Código suavização da onda retificada

```
EMG_envol = Sinal_emgA3_mv
nyq = 0.5 * SamplingRate
low = 300 / nyq
b1, a1 = butter(2, low, btype='low')
EMG_envol = signal.filtfilt(b1, a1, EMG_envol)

EMG_envol_retif=np.abs(EMG_envol)


window_size = 500

 # Criar o filtro de média móvel
filter_window = np.ones(window_size) / window_size

# Aplicar o filtro de média móvel ao sinal retificado
EMG_envol_suave = np.convolve(EMG_envol_retif, filter_window, mode='same')
```

Fonte: Elaborado pelo Autor.

B4 – Comparador

```
media = np.mean(EMG_envol_suave)

binary_signal = []
for i in range(0, len(EMG_envol)):
    if EMG_envol_suave[i] >= (media*0.7):
        binary_signal.append(1)
    else:
        binary_signal.append(0)
```

Fonte: Elaborado pelo Autor

APÊNDICE C – TRECHOS DE CÓDIGO REALIZADO NO PYTHON COM O QT DESIGNER

C1 – Código inicial supervisório com o Qt Designer

```
from PyQt5 import uic, QtCore, QtGui, QtWidgets
from pyqtgraph import PlotWidget
import bitalino
import numpy as np
import time

macAddress = "20:18:06:13:03:69"

device = bitalino.BITalino(macAddress)
SamplingRate=1000    # taxa de amostragem de 1000Hz
device.battery(0)


sinal_analógico = [0, 1023]
faixa_tensão = [-1.65, 1.65]   # tensão em milivolts
variação = faixa_tensão[1] - faixa_tensão[0] # 1,65 -(-1,65) = 3,3mV

running_time = 15
start = time.time()
end = time.time()

def funcao_principal():

    if supervisorio.radioButton.isChecked():
        opcao = "Aquisição: Sinal ACC 2 CANAIS"
    elif supervisorio.radioButton_2.isChecked():
        opcao = "Aquisição: Sinal ACC 1 CANAIS"
    elif supervisorio.radioButton_3.isChecked():
        opcao = "Aquisição: Sinal EMG E EMG 6 CANAIS"
    elif supervisorio.radioButton_4.isChecked():
        opcao = "Aquisição: Sinal EMG 4 CANAIS"
    elif supervisorio.radioButton_5.isChecked():
        opcao = "Aquisição: Sinal EMG 1 CANAL"
    elif supervisorio.radioButton_6.isChecked():
        opcao = "Aquisição: Sinal ACC E EMG 2 CANAIS"
        opcao = "Aquisição: Sinal ACC 1 CANAIS"
        device.start(SamplingRate, [0,4])# Sensor 4(A5)
        data = device.read(3000)
        Sinal_acc_mv = variação*data[:,5]/sinal_analógico[1] + faixa_tensão[0]
        print(Sinal_acc_mv)
        supervisorio.graphicsView_4.plot(Sinal_acc_mv)
        Sinal_emg_mv = variação*data[:,6]/sinal_analógico[1] + faixa_tensão[0]
        print(Sinal_emg_mv)
        supervisorio.graphicsView_2.plot(Sinal_emg_mv)
        device.stop()

    else :
        opcao = ""

    supervisorio.label_6.setText(opcao)

app=QtWidgets.QApplication([])
supervisorio=uic.loadUi("Supervisório-Versão2.ui")
supervisorio.pushButton.clicked.connect(funcao_principal) # Quando clicar INICIAR CHAMA FUNÇÃO PRINCIPAL
supervisorio.show()
app.exec()   # Executa aplicação
```

Fonte: Elaborado pelo Autor.